Bering-uClibc User's Guide

by Bering-uClibc users community and Bering-uClibc Team

Published 2003-12-04

Table of Contents

List of Tables

Chapter 1. Structure of the document

Eric de Thouars <dorus at users.sourceforge.net>

Revision History
Revision 0.2 2003-08-17 ET
Moved Changelog to separate chapter
Revision 0.1 2003-08-11 ET
Initial version

Overview

The LEAF "Bering-uClibc" user's is intended as a guide for Bering-uClibc specific issues. For issues which are not described here, the reader is referred to the Bering User's Guide [http://leaf.sourceforge.net/doc/guide/busers.html]. A lot of the information in that document is directly applicable to Bering-uClibc.

Users contributions are encouraged and welcomed. They can be send to the authors either in plain ASCII form or - better - in Docbook XML format. The XML source code of all chapters is available to everyone and can be used as templates.

Basic prior knowledge of linux and of the LEAF Bering-uClibc distro (or any other LEAF distributions like Bering, Dachstein or Oxygen) is assumed. In particular the reader is supposed to be able to perform the following tasks:

- Add or remove a package to/from a LEAF distribution through editing of the floppy `lrpkg.cfg` file and move it to (out of) the Bering-uClibc floppy disk
- Add or remove a Bering-uClibc linux kernel module by moving it to (out of) `/lib/modules` or `/boot/lib/modules` directory
- Adjust the parameters of a given package through the LEAF configuration menu and backup a package

The following reference is a prerequisite reading:

- The Bering-uClibc Installation guide [http://leaf.sourceforge.net/doc/guide/buc-install.html]
- The Bering Installation guide [http://leaf.sourceforge.net/doc/guide/binstall.html]
- The Bering User's guide [http://leaf.sf.net/doc/guide/busers.html]

Contributions and Feedback

Contributions to and comments on this document can be sent to the Bering-uClibc Team:

K.P. Kirchdoerfer - <kapeka at epost.de>
E. Spakman - <e.spakman at inter.nl.net>
L. Correia - <lfcorreia at users.sourceforge.net>
A. Bernin - <arne at alamut.de>
M. Hejl - <martin at hejl.de>

E. de Thouars - <dorus at users.sourceforge.net>

Tip

You can download the docbook xml sources from the different sections of this user's guide here [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/leaf/doc/guide/user-bering-uclibc/] to be used as a template. A complete Docbook XML documentation can be found here [http://www.docbook.org/tdg/en/html/docbook.html].

Chapter 2. Using Dropbear

Matt Johnston <matt at ucc.asn.au>
K.P. Kirchdoerfer <kapeka at epost.de>
Eric de Thouars <dorus at users.sourceforge.net>

Revision History
Revision 0.1 2003-08-11 ET
Initial version

Objectives

This chapter describes the initial installation and configuration of the light weight ssh server "Dropbear" which is part of the base Bering-uClibc distribution.

Dropbear was developed by Matt Johnston and for more information on Dropbear itself you should visit his webpages [http://matt.ucc.asn.au/dropbear/dropbear.html].

Note

Export of cryptographic software from Australia is subject to export controls - you should ensure that you are not breaching these controls. See Crypto Law Survey [http://rechten.kub.nl/koops/cryptolaw/] for some good research.

Comments on this chapter should be addressed to its maintainer: Eric de Thouars <dorus at users.sourceforge.net>.

Step 1: Load the dropbear package

Note

For Bering-uClibc, dropbear and dropbearkey have been compiled into one binary, just like busybox that also provides different applications in one binary. Therefore only one package (`dropbear.lrp`) is needed. This is a difference from other ssh applications (sshd, lshd) used with LEAF packages, where key generation utility and daemon are provided in two separate packages.

If you start with a fresh Bering-uClibc image you can skip this step because the default `leaf.cfg` file provided with Bering-uClibc looks like this:

```
LRP="root config etc local modules iptables dhcpcd keyboard shorwall ulogd dnsmasq
```

The package `dropbear.lrp` is loaded on startup.

If you have edited leaf.cfg in the past, and `dropbear.lrp` is currently not installed on your system, you can do two things:

- add the package again to `leaf.cfg` and reboot (Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-lrpkg.html] to learn how to do that.)
- add `dropbear.lrp` to lrpkg.cfg/leaf.cfg and load package manually.

Step 2: Generate the keys

The keys necessary for the ssh server can be generated with the command **gendropbearkeys**. After giving this command, sit back and enjoy a cup of coffee while your machine generates the RSA and DSS keys.

tip: use weblet to generate entropy.

Note

Backup the `dropbear.lrp` package to save the keys

Step 3: Set root password

Dropbear will not let you log in as "root" without a password. Set the root password with the command **passwd** while logged in as "root".

Note

Backup the `etc.lrp` package

Step 4: Check Shorewall rules

The default configuration of the Shorewall package provided with Bering-uClibc should allow you to login to your LEAF box with ssh from the local network. Nevertheless it is wise to make sure that this is really so.

Assuming that you have not renamed the zone for the local network, this zone is called "loc". The file `/etc/shorewall/rules` should then have lines like this:

```
#############################################################################
#ACTION  SOURCE         DEST            PROTO   DEST    SOURCE     ORIGINAL
#                                               PORT    PORT(S)    DEST
(...)
# Accept SSH connections from the local network for administration
#
ACCEPT   loc            fw              tcp     22
(...)
```

If this is not the case, add these lines and backup the `shorwall.lrp` package.

Step 5: Finishing up

Reboot your machine and watch dropbear start. You can now remotely log in to your Bering-uClibc box with an ssh client or scp files from/to your Bering-uClibc box.

Miscellaneous

Note that you can't run dropbear and sshd at the same time, unless you change dropbear or sshd's port. / etc/default/dropbear is the config file for dropbear.

Chapter 3. Using dnsmasq

K.-P. Kirchdörfer <kapeka at users.sourceforge.net>

Revision History
Revision 0.1 2004-06-03 kp
Initial Document

Objectives

Dnsmasq is a lightweight, easy to configure DNS forwarder and DHCP server. It is designed to provide DNS and optionally, DHCP, to a small network. It can serve the names of local machines which are not in the global DNS. The DHCP server integrates with the DNS server and allows machines with DHCP-allocated addresses to appear in the DNS with names configured either in each host or in a central configuration file.

Dnsmasq supports static and dynamic DHCP leases and BOOTP for network booting of diskless machines.

An almost complete feature list can be found on the author's page. [http://thekelleys.org.uk/dnsmasq/doc.html]

The configuration documentation is contained in the configuration file `/etc/dnsmasq.conf`.

Here you'll find a few hints how to get a basic configuration of dnsmasq done. It is advised that you read the configuration file carefully, to get most out this application.

Beginning with Bering-uClibc 2.2 dnsmasq will replace dnscache on the base image. Additionally it adds features previously only available if both dhcpd and tinydns were loaded.

It will still be possible for users to switch back and use dnscache, dhcpd and tinydns.

Load dnsmasq package

If you are using Bering-uClibc 2.2 or higher, this step can be skipped.

For older Bering-uClibc versions edit `lrpkg.cfg` and add `dnsmasq` to packages list:

```
root,config,etc,local,modules,iptables,keyboard,shorwall,ulogd,dnsmasq ..."
```

And you can remove dnscache from `lrpkg.cfg`, because it's replaced by dnsmasq.

Configure dnsmasq dns forwarder

dnsmasq works with various sources to provide resolving domain names on your local network. It is capable of using /etc/hosts, /etc/resolv.conf, additional resolv.conf files created by other applications like ppp, acting as secondary DNS in addition to primary DNS and is well integrated with the dhcpd part of dnsmasq.

Again we advise you to read the configuration file carefully, to understand how dnsmasq integrates into your network. We will describe a few standard settings for a basic LEAF image setup.

The first decision you have to make, is wether you like to use your own resolv.conf, or one created by an

another application (see below).

```
# Change this line if you want dns to get its upstream servers from
# somewhere other that /etc/resolv.conf
#resolv-file=
```

In case you use your own /etc/resolv.conf, leave this as is.

If you want dnsmasq to resolve your local and private domain as well (either from /etc/hosts or dhcp) set your domain as local

```
# Add local-only domains here, queries in these domains are answered
# from /etc/hosts or DHCP only.
local=/private.network/
```

Next choose the interface(s) dnsmasq should listen - the one connected your LAN. In a simple LEAF setup it is usually eth1.

```
# If you want dnsmasq to listen for requests only on specified interfaces
# (and the loopback) give the name of the interface (eg eth0) here.
# Repeat the line for more than one interface.
interface=eth1
```

If you have more than one interface connected to local LAN's you may define the interface *not* to listen on - the interface to the Internet:

```
# Or you can specify which interface _not_ to listen on
except-interface=eth0
```

At last you should configure to expand hostnames in your LAN and your domain:

```
# Set this (and domain: see below) if you want to have a domain
# automatically added to simple names in a hosts-file.
expand-hosts

# Set the domain for dnsmasq. this is optional, but if it is set, it
# does the following things.
# 1) Allows DHCP hosts to have fully qualified domain names, as long
#     as the domain part matches this setting.
# 2) Sets the "domain" DHCP option thereby potentially setting the
#    domain of all systems configured by DHCP
# 3) Provides the domain part for "expand-hosts"
domain=private.network
```

For debugging purposes you can enable "log-queries" at the end of dnsmasq.conf.

Now you're nearly done with a default setup. Read on in one of the following section best describing your Internet connection.

Configure dnsmasq dhcpd

The integrated DHCP server dhcpd is disabled by default.

To enable it supply the range of addresses available for lease, and optionally a lease time:

```
# Uncomment this to enable the integrated DHCP server, you need
# to supply the range of addresses available for lease and optionally
# a lease time. If you have more than one network, you will need to
# repeat this for each network on which you want to supply DHCP
# service.
dhcp-range=192.168.1.1,192.168.1.199,12h
```

dnsmasq supports various methods setting fixed ip's in your LAN, e.g. by name, MAC adress.

dnsmasq integrated DHCP server also supports sending options to the hosts asking for a lease as described in RFC2132 [http://www.faqs.org/rfcs/rfc2132.html]. For the common setting (subnet mask, default router, DNS server and broadcast address) dnsmasq sets sane defaults.

Using dnsmasq with ppp/pppoe

pppd (and so pppoe) is capable to receive the upstream nameservers from your provider during connect and store them in `/etc/ppp/resolv.conf`.

To enable that feature you have to set the option `usepeerdns` either in `/etc/ppp/peers/dsl-provider` or `/etc/ppp/options`.

Next you have to change/enable dnsmasq to use that resolv.conf (probably additionally to /etc/hosts).

Edit `/etc/dnsmasq.conf` and set the `resolv-file`:

```
# Change this line if you want dns to get its upstream servers from
# somewhere other that /etc/resolv.conf
resolv-file=/etc/ppp/resolv.conf
```

Note

Backup dnsmasq.lrp and ppp.lrp before reboot.

Using dnsmasq with dhcpcd

dhcpd gets upstream DNS servers while connecting to your ISP and stores them in `/etc/dhcpc/resolv.conf`.

Edit `/etc/dnsmasq.conf` file and point to the `/etc/dhcpc/resolv.conf` file.

```
# Change this line if you want dns to get its upstream servers from
# somewhere other that /etc/resolv.conf
resolv-file=/etc/dhcpc/resolv.conf
```

Note

Backup dnsmasq.lrp reboot.

Using dnsmasq with static ip

Edit `/etc/resolv.conf` and add the upstream DNS servers. There is no extra configuration needed for dnsmasq.

Note

Backup etc.lrp before reboot.

Using dnsmasq with pump

Remove "nodns" in `pump.conf` to let pump update/overwrite `/etc/resolv.conf`.

Note

Backup pump.lrp before reboot.

Chapter 4. Using Bering-uClibc with an IDE harddisk or CD-ROM drive

K.-P. Kirchdörfer <kapeka at user.sourceforge.net>
Peter Mueller <peter.nospam.anarchy.com>
Luis F. Correia <lfcorreia at user.sourceforge.net>
Eric de Thouars <dorus at user.sourceforge.net>
Jacques Nilo <jnilo at users.sourceforge.net>
Eric Wolzak <ericw at users.sourceforge.net>

Revision History
Revision 0.1 2003-10-30 kp
Initial version
Revision 0.2 2003-11-06 kp
Additional inks for CD building
Revision 0.3 2003-11-19 et
PXEBoot chapter
Revision 0.4 2004-05-27 kp
added IDE chapter (from Bering Guide)
Revision 0.5 2004-06-24 kp
reworked IDE chapter
Revision 0.6 2004-09-06 kp
added IDE-CF chapter written by Peter Mueller

initrd.lrp

To boot from a IDE-based medium you need to add the ide-related modules and hd/cd-rom modules to initrd.lrp (/boot/lib/modules) and to modify /boot/etc/modules. For your convenience the Bering-uClibc team provides an already enhanced initrd.lrp with all modules needed to boot from an IDE harddisk or IDE CD-ROM drive for Bering-uClibc version 2.0 and above.

You can download the file initrd_ide_cd.lrp [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/leaf/bin/packages/uclibc-0.9/20/initrd_ide_cd.lrp?rev=HEAD&content-type=application/octet-stream] from CVS.

Please rename initrd_ide_cd.lrp to initrd.lrp and use this initrd.lrp while following the instructions below.

Create a bootable CD-ROM

Introduction

To create a bootable CD-ROM you may follow the instructions in the Bering User's Guide Chapter 10, with the exception that you don't have to build a new initrd if you use `initrd_ide_cd.lrp` renamed to `initrd.lrp`.

The approach described in the Bering User's Guide has the disadvantage that due to bad BIOS implementations, the CD may not boot in older computers.

We will describe the more general approach the Dachstein versions used to create a bootable CD-ROM,

which allows to boot from CD on every computer that is able to boot from CD.

Main trick is to provide a bootable 1,44Mb floppy diskimage on the CD-ROM.

Step1 Create a bootable 1,44MB floppy

Format a floppy disk, add a msdos filesystem and make it bootable with syslinux:

```
# fdformat /dev/fd0
# mkfs.msdos /dev/fd0
# syslinux -sf /dev/fd0
```

Now mount the floppy and copy the kernel (linux), syslinux.cfg and syslinux.dpy from the Bering-uClibc diskimage onto the floppy. Copy `initrd_ide_cd.lrp` renamed to `initrd.lrp` onto the floppy.

Before umounting the floppy edit syslinux.cfg on the floppy disk.

syslinux.cfg for Bering-uClibc 2.1 and earlier versions

Edit `syslinux.cfg` and make shure the PKGPATH points to the CD-ROM device and floppy as well (that's the place where you store your configuration settings).

```
display syslinux.dpy
timeout 0
default linux initrd=initrd.lrp init=/linuxrc rw root=/dev/ram0 boot=/dev/fd0:msdo
PATH=/dev/cdrom:iso9660,/dev/fd0:msdos LRP=root,etc,loca.....
```

Now you have a bootable floppy for your CD.

syslinuxcfg for Bering-uClibc 2.2 and later versions

Edit `syslinux.cfg` and change the LEAFCFG variable to point to your floppy device (so you can easily add or remove packages to load without buring a new ISO-image:

```
display syslinux.dpy
timeout 0
default linux initrd=initrd.lrp init=/linuxrc rw root=/dev/ram0 LEAFCFG=/dev/fd0:m
```

This will be your bootable the floppy for the CD creation.

The floppy device will be used to store your configuration settings.

Step 2: Create the CD

Now you are ready to build your CD-ROM. Create a new directory and put all packages you like to have available on your CD into it.

Next dump your boot floppy build above into the same directory.

```
# dd if=/dev/fd0 of=bootdisk.ima bs=8k
```

Create an ISO-Image from that directory and burn it.

```
# mkisofs -v -b bootdisk.ima -c boot.catalog -r -J -f -o Bering-uClibc-CD.iso
# cdrecord -v dev=[target] Bering-uClibc-CD.iso
```

Step 3: Adding packages and backup configuration

Packages can be added or removed in a flexibel way by declaring/undeclaring them in `lrpkg.cfg` (Bering-uClibc <= 2.1) or `leaf.cfg` (Bering-uClibc >= 2.2) on a new formatted floppy.

Additionally your configuration settings for all packages can be stored on the same floppy.

Declaring packages for Bering-uClibc 2.1 and earlier versions

To add or remove packages just edit `lrpkg.cfg` on a blank formatted floppy disk - all entries on one line. It looks like:

```
root,etc,local,modules,pump,keyboard,shorwall,dnscache,weblet
```

Declaring packages for Bering-uClibc 2.2 and later versions

Edit `leaf.cfg` on a blank formatted floppy disk, add your packages to LRP and change PKGPATH to point to your CDROM and the floppy device.

```
LRP="root config etc local modules iptables dnsmasq keyboard shorwall ulogd libz m
PKGPATH=/dev/fd0:msdos,/dev/cdrom:iso9660
syst_size=8M
log_size=2M
```

Note

The order in PKGPATH is important!

The leftmost entry will be loaded last - so your packages will be load first from CDROM and then from /dev/fd0. This will overwrite the configuration with the settings you stored on the floppy.

Backing up your configuration

You can backup your configuration changes onto the floppy, you have declared `leaf.cfg`/`lrpkg.cfg`.

To only backup the changes in configuration and not the complete packages, which may be too big to fit onto a floppy, choose "partial backup" and /dev/fd0 as destination for the packages.

Important

Partial backup does not work for etc.lrp.

Create a bootable HD

To install Bering-uClibc on an IDE device, proceed as follows:

You have to make sure your IDE device has a first bootable partition and is DOS formatted.

Warning

Be careful: you will be destroying any pre-existing data !

Replace initrd.lrp on your Bering-uClibc floppy with initrd_ide_cd.lrp [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/leaf/bin/packages/uclibc-0.9/20/initrd_ide_cd.lrp?rev=HEAD&content-type=application/octet-stream] and boot from that floppy.

Keep a second floppy with the hdsupp.lrp package around and insert this floppy after boot.

After login mount the new floppy with hdsupp.lrp, install hdsupp.lrp, partition and format your IDE disk:

```
mount /dev/fd0 /mnt
cp /mnt/hdsupp.lrp /
cd /
lprkg -i hdsupp
fdisk /dev/hda
```

Create an empty DOS partition table (using the 'o' command), create a primary partition and make that bootable (using the 'a' command). Save your changes with 'w'.

Format the IDE device:

```
mkfs.msdos /dev/hda1
```

and create a Master Boot Record:

```
dd if=/usr/sbin/mbr.bin of=/dev/hda bs=512 count=1
```

Now you can install syslinux; issue the following command:

```
syslinux [-s] /dev/hda1
```

The -s flag might be required for syslinux to work with old buggy BIOSes. See the syslinux [http://syslinux.zytor.com/faq.php] web site for more instructions.

Umount the floppy with hdsupp.lrp, reinsert your boot floppy and mount it:

```
mount /dev/fd0u1680 /mnt
```

Once this is done, the steps for Bering-uClibc 2.1 and previous versions and Bering-uClibc 2.2 and later versions are different.

Bering-uClibc 2.1 and earlier versions

For Bering-uClibc 2.1 and earlier versions edit the `syslinux.cfg` file in /mnt and change the "boot" and "PKGPATH" entries to point to your harddisk. It will look like:

```
display syslinux.dpy
timeout 0
default linux initrd=initrd.lrp init=/linuxrc rw root=/dev/ram0 boot=/dev/hda1:msd
```

```
LRP=root,etc,loc
```

Once you have finished with your floppy preparation, copy all the files from it except `ldlinux.sys` (which is created by syslinux) to the IDE device that you prepared earlier. You should now be able to boot from the IDE device. Once again be careful not to copy `ldlinux.sys` from the floppy otherwise your disk won't be bootable and you will have to go over the installation of syslinux on your hard disk again.

An alternative methode is to prepare and load the packages onto your disk with pxeinstall.tgz described in the next chapter. Besides it's pretty fast, once you've setup the environment, it is especially useful if your router has no floppy drive.

Bering-uClibc 2.2 and later versions

For Bering-uClibc 2.2 and later versions edit `syslinux.cfg` and change the LEAFCFG variable to point to your harddisk:

```
display syslinux.dpy
timeout 0
default linux initrd=initrd.lrp init=/linuxrc rw root=/dev/ram0 LEAFCFG=/dev/hda1:
```

Edit `leaf.cfg` and add your packages to LRP and change PKGPATH to point to your harddisk.

```
LRP="root config etc local modules iptables dnsmasq keyboard shorwall ulogd libz m
PKGPATH=/dev/hda1:msdos
syst_size=8M
log_size=2M
```

If you have declared two partitions on your harddisk, and intend to use the second partition for backups, you have to add the second partition in the PKGPATH variable:

```
LRP="root config etc local modules iptables dnsmasq keyboard shorwall ulogd libz m
PKGPATH=/dev/hda2:msdos,/dev/hda1:msdos
syst_size=8M
log_size=2M
```

Note

The order in PKGPATH is important!

Your second partition has to be the first entry to load the stored configuration, or partial backup, after the original (unconfigured) package.

Once you have finished with your floppy preparation, copy all the files from it EXCEPT `ld-linux.sys` (which is created by syslinux) to the IDE device that you prepared earlier. You should now be able to boot from the IDE device. Once again be careful not to copy `ldlinux.sys` from the floppy otherwise your disk won't be bootable and you will have to go over the installation of syslinux on your hard disk again.

An alternative methode is to prepare and load the packages onto your disk with pxeinstall.tgz described in the next chapter. Besides it's pretty fast, once you've setup the environment, it is useful if your router has no floppy drive.

Using pxeinstall.tgz

Introduction

This section describes how to setup an environment in which you can use PXE to boot systems like a Soekris [http://www.soekris.com] net4501 or a LexSystem [http://www.lex.com.tw/index1.htm] (tested with a CV860A version). A lot of information on this issue is already available on the internet. A lot of these pages describe how to get a system with complete functionality up and running this way. The main focus of this chapter is to get the system up and running with PXE boot so that you can format the CF card, put an msdos file system and syslinux on the CF card and finally copy all files necessary for a Bering(-uClibc) system on the CF card.

With the functionality of pxeinstall.tgz you can:

- boot the system via the network,
- put a number of different file systems on the CF card (minix, ext2, swap, msdos),
- make the CF card bootable (syslinux) and
- download files onto the card via http or ftp (wget)
- download files onto the card via Windows networking (smbmount and cp)

If you need more functionality than this, you'll need to search further .....

Caution

The PXE boot functionality can only be used to get a system up and running, it is not intended to be used for regular use. For example, you cannot backup any files because the backup function does not exist.

Requirements

- A system up and running with a DHCP and a TFTP server. I used a RedHat 9.0 system with standard dhcp and tftp servers (not the pxeboot server that comes with RH 9.0).

  ### Caution

  - The DHCP and the TFTP server must be running on the same ip address to get PXE boot working !!
  - The TFTP server MUST support the *tsize* option. The LexSystem is known to reboot without proper error messages if a TFTP server is used that doesn't support the *tsize* option.

- The pxeinstall.tgz [http://cvs.sourceforge.net/viewcvs.py/leaf/bin/packages/nolibc/pxeinstall.tgz?rev=HEAD&content-type=application/octet-stream] tarball which contains the files that must be put on the TFTP server.
- Systems that do not have a keyboard and videocard (such as the Soekris) also need a system connected to the serial port so that you can control the system.

General description of the PXE boot sequence

The pxeboot sequence goes as follows:

1. BIOS starts
   - The necessary IP addresses are acquired via DHCP. (IP address and the IP address of the TFTP server).
   - The pxelinux.0 file is downloaded from server via TFTP. (pxelinux.0 is a network boot loader).
   - The pxeconfig file is downloaded from server via TFTP. The pxeconfig file looks very much like the syslinux.conf file for normal LEAF booting. One of the additions is that some information is passed to the kernel command line for IP autoconfiguration at kernel load time (see step 3)
2. PXELinux starts
   - The kernel is downloaded from server via TFTP from the location specified in the pxeconfig file
   - The initrd file is downloaded from server via TFTP from the location specified in the pxeconfig file
3. Kernel starts
   - The network interface is initialized and autoconfigured using the parameters in pxelinux config file
4. Initrd starts
   - The initrd image contains TFTP client which is used to download the packages from the TFTP server.
   - A modified linuxrc downloads the packages supplied in the LRP variable from the TFTP server address mentioned in the "boot" variable
5. Normal leaf boot sequence continues from here. Packages are uncompressed and untarred and the system starts.

Configuration

DHCP server

To configure the DHCP server you need to find out the MAC address of the interface on which the PXE boot will take place. In most cases the MAC address is shown when the PXE client in the BIOS starts.

See the system specific sections on Soekris and LexSystem how to find out the right MAC address on your system.

When you have the address, edit the file `/etc/dhcpd.conf`:

```
#
subnet 192.168.1.0 netmask 255.255.255.0 {

    default-lease-time 600;
    max-lease-time 7200;
```

```
    host pxe {
        hardware ethernet  00:00:c3:2f:63:80;
        fixed-address 192.168.1.254;
        option host-name "pxe";
        filename "pxelinux.0";
    }
}
```

Restart the dhcp daemon.

TFTP server

Unzip and untar the pxeinstall.tgz file in the root directory of the TFTP server. On my system this is `/tftpboot`.

The directory `/tftpboot/pxelinux.cfg` contains three files: `default`, `lexsystem` and `net4501`. The file `default` is the one being used by the PXE boot functionality and is right after unzipping and untarring a copy of the `lexsystem` file.

This file has pretty much the same layout as any other syslinux.cfg file and defines where the kernel and the initial file system image can be found. Like any other LEAF distribution it also contains the packages to be installed. In this specific case the packages will be downloaded with TFTP prior to installation.

Depending on the system that you want to boot via PXE you should copy either the `lexsystem` file or the `net4501` file to `default`. The `lexsystem` file can be used for systems with a keyboard and video card. The `net4501` file should be used for systems with only a serial console.

Now that the needed servers are configured it is time to go to your specific system. In the following sections, the PXE boot sequence for the Soekris system is described, the one for LexSystem is similar.

Booting via PXE

Soekris

Connect a terminal to the serial port and fire-up your Soekris system. You should see something like this:

```
OS ver. 1.10  20020603  Copyright (C) 2000-2002 Soekris Engineering.

Soekris Engineering net4501               CPU 80486 134 Mhz

0064 Mbyte Memory

PXE-M00: BootManage UNDI, PXE-2.0 (build 082)

Slot   Vend Dev  ClassRev Cmd  Stat CL LT HT  Base1    Base2   Int
-------------------------------------------------------------------
0:00:0 1022 3000 06000000 0006 2280 00 00 00 00000000 00000000 00
0:18:0 100B 0020 02000000 0107 0290 00 3F 00 0000E001 A0000000 10
0:19:0 100B 0020 02000000 0107 0290 00 3F 00 0000E101 A0001000 11
0:20:0 100B 0020 02000000 0107 0290 00 3F 00 0000E201 A0002000 05

 5 Seconds to automatic boot.   Press Ctrl-P for entering Monitor.
```

Caution

The Soekris only supports PXE boot via the NET0 interface. So make sure that your NET0 interface and the DHCP/TFTP server are connected to the same network !!

Now press **<Ctrl-P>** and give the command **boot f0**.

```
comBIOS Monitor.   Press ? for help.

> boot F0

BootManage UNDI, PXE-2.0 (build 082)
BootManage PXE-2.0 PROM 1.0, NATSEC 1.0, SDK 3.0/082 (OEM52)
Copyright (C) 1989,2000 bootix Technology GmbH, D-41466 Neuss.
PXE Software Copyright (C) 1997, 1998, 1999, 2000 Intel Corporation.
Licensed to National Semiconductor

CLIENT MAC ADDR: 00 00 C3 2F 63 80
```

Here you have the MAC address that you need to configure your DHCP server. If your DHCP and TFTP server were correctly setup and are connected to the right interface of the Soekris, the boot sequence should continue with:

```
DHCP /

CLIENT IP: 192.168.1.254  MASK: 255.255.255.0  DHCP IP: 192.168.1.200

TFTP /

PXELINUX 1.76 2002-08-27  Copyright (C) 1994-2002 H. Peter Anvin

Linux version 2.4.18 ..............
..
<snip> Linux kernel loading
..
Kernel command line: console=ttyS0,19200 BOOT_IMAGE=pxe/linux ip=192.168.1.254:::2
..
eth0: link up.
IP-Config: Complete:
      device=eth0, addr=192.168.1.254, mask=255.255.255.0, gw=255.255.255.255,
     host=pxe, domain=, nis-domain=(none),
     bootserver=255.255.255.255, rootserver=255.255.255.255, rootpath=
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
..
<snip> kernel loading continues
..
```

As you see, during kernel loading the ip configuration is set based upon the parameters passed on the kernel command line.

The whole sequence should end with a login prompt. You what to do next ;-)

LexSystem

The LexSystem we have tested is based on the so-called CV860A board with a VIA C3 533A processor. The board supports up to 512MB PC133 SDRAM and is delivered with two or optional three network interfaces (usually Realtek with rtl8139too driver). Mass storage devices supported are IDE HD, CF and

DOM.

It is a good idea to have LAN as third boot device in the "Advanced BIOS Features". The network interfaces can be configured by pressing **<Shift-F10>** to enter the NIC BIOS setup. The options that must be set here are:

- Network Boot Protocol: PXE
- Boot Order: Int 18h (boot the devices ordered in Bios Setup)
- show config message and show message time do not really matter...

One of the problems of the LexSystem is, that you do can not recognize which NIC you are configuring as it is not really shown if you don't have all 3 messages enabled (3 NIC boot agent configuration roms out there). Also the order in the BIOS is not the order of the interfaces set by the linux kernel...

We had to provide an separate configuration file, because the board behaves somewhat wierd during setup/installation with pxeinstall. During pxe part of booting it uses eth0 and after getting a dhcp address and changing to TFTP to load kernel, basic cfg and basic applications it uses eth1. Additionally it is important that eth0 and eth1 connect the same LAN segment during install, because DHCP server and TFTP server has to be accessible on the same IP address.

Setting up the new system

If all went well, you should now be looking at a login prompt on your system. Login as "root", no password is required. The CF card can be formatted and installed with syslinux with the following commands:

```
pxe: -root-
# mkdosfs /dev/hda1
mkdosfs 0.3b (Yggdrasil), 5th May 1995 for MS-DOS FS

pxe: -root-
# syslinux /dev/hda1

pxe: -root -
# dd if=/usr/sbin/mbr.bin of=/dev/hda bs=512 count=1
```

The last command installs a master boot record on to your IDE disk.

Now you can use the **wget** command to download all the files you need to the CF card. Another option is to use **smbmount** to mount a Windows share to `/mnt` and copy all necessary files.

Supported network cards

The pxeinstall.tgz requires that all supported network cards have to be compiled into the kernel, kernel modules for network cards are not supported. To allow you to make use of pxeinstall we added at least all modules provided with LEAF Bering-uClibc 2.0 to this special kernel . Please let us know, if you have success with hardware and network interface cards other than tested and described in this document.

Currently supported/compiled into the kernel are:

- 3c590/3c900 series (592/595/597) "Vortex/Boomerang"

- AT1700/1720
- AMD PCnet32 PCI
- DECchip Tulip (dc21x4x) PCI
- EtherExpressPro/100
- National Semiconductor DP8381x series PCI Ethernet
- PCI NE2000 and clones
- RealTek RTL-8139 PCI Fast Ethernet Adapter
- SMC EtherPower II
- VIA Rhine
- Winbond W89c840 EthernetI

Create a bootable IDE-CF

This section is a contribution by Peter Mueller and describes how create a bootable IDE-CF device.

Booting from an onboard IDE-CF system

You can purchase CF-IDE adapters for very cheap. Both parts can be purchased for $30 US or less. The setup is simple. 1.) Setup the CF flash in the system. Note that you will want to configure the IDE CF card manually instead of letting IDE auto-detect the settings. To find the setting for your CF card, use IDEINFO [http://www.tech-pro.net/ideinfo.html]. If you auto-configure the CF you might have big problems!!

2.) Create a dos bootdisk floppy from bootdisk.com. I have used http://csislabs.palomar.edu/Student/Utilities/boot622.exe succesfully.

3.) Install syslinux.com onto the floppy. The file is available from http://www.kernel.org/pub/linux/utils/boot/syslinux/. Grab the .zip file and extract syslinux.com onto the floppy that you just made.

Note

If you run into problems with latest version you may want to use syslinux 2.07, which has been proofed to work.

4.) Boot from the floppy on the IDE-CF system. Fdisk the drive. If there are any partitions on the drive, delete them and reboot before proceeding further. Setup a primary DOS partition,and make it active. Reboot.

5.) Boot from the floppy again. Format the CF card with "**format c:**".

Note

If you have other IDE devices in the system, the CF card might not be C:. Be careful here!

6.) After the format is complete, run "**syslinux -s c:**".

7.) Download the latest Bering-uClibc image.

8.) Using a CD-R, sneakernet (floppy), "CF on another machine", or whatever means you feel comfortable with, transfer the Bering-uClibc LRP & txt files to the floppy. Do NOT transfer `ldlinux.sys` or you will have to start over.

9.) Change the syslinux.cfg part "LEAFCFG=/dev/fd0:msdos" to "LEAFCFG=/dev/hda1:msdos".

10.) Change the leaf.cfg part 'PKGPATH="/dev/fd0:msdos"' to 'PKGPATH="/dev/hda1:msdos"'

11.) Install `initrd.lrp` with IDE support instead of standard initrd.lrp Currently this package is http://leaf.sourceforge.net/packages/uclibc-0.9/20/initrd_ide_cd.lrp Rename the package to initrd.lrp and install on the CF card.

12.) Reboot & configure your happy IDE-CF system.

Booting from a PCI-IDE CF system

Most of the steps are the same. You will need to ask the Bering-uClibc team for a kernel that supports your add-on card. Additionally, you must turn off DMA support on your device or it will work erratically. Here is how I did it in `syslinux,cfg`:

```
serial 0 19200
display syslinux.dpy
timeout 0
default bzimage initrd=initrd.lrp init=/linuxrc rw root=/dev/ram0 syst_size=20M
 log_size=20M tmpfs_size=256M LEAFCFG=/dev/hda1:msdos
append console=ttyS0,19200 nodma=hda ide=nodma
```

Change hda to whatever your device is. Note the syst_size, log_size, etc. options that you normally see in leaf.cfg. These can be ignored, you can put these in leaf.cfg. I have tried 10 different cards. The only card I have had any success with is the SIIG Ultra-ATA 100. The SIIG Ultra-ATA 133 is a different chipset. Here is the product - http://www.siig.com/product.asp?pid=429. If the link is broken, it is chipset CMD0649 in linux. If you have any choice at all, use onboard IDE. The add-on cards are not worth the pain.

Credits

Thanks to the Bering-uClibc & LEAF teams for a great product! Thanks Nicholas Fong! Your page @ http://chinese-watercolor.com/LRP/hd/ is very nice!

Links

Building a LEAF CD-ROM

Other sources how to build a CD-ROM are:

- Charles Steinkuehler's LRP CD [http://leaf.sourceforge.net/devel/cstein/Packages/LRP-CD.htm] originylly written for LEAF Dachstein version
- Chapter 10 of Bering User's Guide [http://leaf.sourceforge.net/doc/guide/bucdrom.html] written by Luis Correia

Chapter 5. Serial Modem configuration

Jacques Nilo <jnilo at users.sourceforge.net>
Eric Spakman <espakman at users.sourceforge.net>

Revision History		
Revision 0.4	2004-05-04	ES
Update for leaf.cfg		
Revision 0.3	2004-03-06	ES
Update for Bering-uClibc		
Revision 0.2	2002-04-14	JN
corrected and edited		
Revision 0.1	2002-03-15	JN
initial revision		

Objectives

We assume here that you can only get connected to internet through a serial modem connection and that you want to share that connection with other (internal) computers in your home or office. What follows describe the configuration of this dial-up modem router. Your external interface (to the internet) will be ppp0, your internal interface (to your internal network) is supposed to be done through an ethernet network card (eth0).

The PPP-Howto [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html] is a useful reference for this section.

Comments on this section should be addressed to its maintainer: Eric Spakman <espakman at users.sourceforge.net>. Thanks to Lee who provided useful additions to this section.

Bering-uClibc comes with two ppp daemons, one with filter support and one without. The ppp.lrp package on the base image contains the ppp daemon without filter support. The ppp-filter.lrp package can be used for demand-dialing mode and needs the libpcap.lrp package. Before using the filter version, the package needs to be renamed to ppp.lrp.

The ppp source is version 2.4.2 and supports ipv6, mschapv2, mppe and optional pppoe or pppoatm with plugins

Step 1: declare the ppp package

Boot a Bering-uClibc floppy image. Once the LEAF menu appears get access to the linux shell by (q)uitting the menu. Edit the `lrpkg.cfg` (pre Bering-uClibc-2.2.0) or `leaf.cfg` (Bering-uClibc-2.2.0 onwards) file and replace the dhcpcd entry by ppp in the list of packages to be loaded at boot. Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-lrpkg.html] to learn how to do that.

Step 2: declare the ppp modules

In order to have a modem dialup connection working, you need to have ppp support enabled through the appropriate kernel modules. You also need to declare the driver module of the network card assigned to your internal network. In the following example, this card is supposed to be a standard ne 2000 PCI card.

To configure your modules, go to the LEAF Packages configuration menu and choose modules. Enter 1)

to edit the `/etc/modules` file and enter the following information:

```
# 8390 based ethernet cards
8390
ne2k-pci

# Modules needed for PPP connection
slhc
ppp_generic
ppp_async
ppp_deflate

# Masquerading 'helper' modules
ip_conntrack_ftp
ip_conntrack_irc
ip_nat_ftp
ip_nat_irc
```

Important

The sample file above might be different in your own case: you might need another network module or some extra functionnalities. Adjust to your needs !

Backup the modules.lrp package.

Step 3: configure ppp

Connection with your ISP will be handled by PPP. The PPP How-to [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html] document will give you very detailed information about this protocol and how to set-up the numerous parameters.

Through the LEAF packages configuration menu get access to ppp configuration. The following menu will show-up

```
                        ppp configuration files

       1) ISP pppd options
       2) ISP login script
       3) System wide pppd options
       4) chap secret
       5) pap secret

  q) quit
  ------------------------------------------------------------------------------
       Selection:
```

Entry 1) allows you to adjust the parameter of your ppp connection through the `/etc/ppp/peers/provider` file. The most important argument is the *ttySx* parameter which defines the serial port to which your modem is connected.

Tip

Look at your `/var/log/syslog` file after booting Bering-uClibc. It will give you the list of the serial ports recognized by your linux kernel.

A working `/etc/ppp/peers/provider` file for a Compuserve connection could look like:

```
# ISP pppd options file
# What follows is OK for Compuserve
#
noauth
debug           # log transaction to /var/log/messages
/dev/ttyS0      # (ttyS0=com1, ttyS1=com2, ...)
115200          # baud  rate
modem
crtscts         # use hardware flow control
asyncmap 0
defaultroute    # ppp becomes default route to the internet
noipdefault
lock            # don't let other processes besides PPP use the device
connect "/usr/sbin/chat -v -f /etc/chatscripts/provider"
```

If you plan to dial into a Windows RAS server or a server that uses PAP or CHAP authentication, you need to add a line to this file. Just above the "connect" command, on a line of its own, add:

```
name your_ISP_login
connect "/usr/sbin/chat -v -f /etc/chatscripts/provider"
```

You need this because ppp has to masquerade the firewall as you when using PAP or CHAP authentication.

Entry 2) allows you to adjust the communication script which will handle the connection with your ISP. This script is stored in the `/etc/chatscripts/provider`

A working script for a Compuserve connection could look like:

```
# ISP login script
# What follows is OK for Compuserve
# Adjust to your taste
ABORT "BUSY"
ABORT "NO CARRIER"
ABORT "VOICE"
ABORT "NO DIALTONE"
ABORT "NO ANSWER"
"" ATZ
# ISP telephone number: 124567890
OK ATDT1234567890#
CONNECT ''
Name: CIS
# With compuserve your_login_account=12345,6789
ID: your_login_account/go:pppconnect
Password: your_password
PPP
```

If you are not using Compuserve you should also delete all of the lines below the <CONNECT "> line. A few - very few - ISPs require the final "PPP" line these days.

Edit Entry 3) - /etc/ppp/options "System-wide pppd options" if you want the system to demand dial and to drop the line if idle for a preset time. To do this, change "persist" to "demand" and add another line below "demand" that says "idle 600", where 600 is the number of seconds the system should wait before dropping hanging up if there is no network traffic.

Edit either the PAP (Entry 4) or CHAP (Entry 5) option to set up how your system authenticates. For

PAP authentication, choose the PAP option and add a line giving your ISP login and password. Your ISP login must be the same antry as the one provided after the name entry in your ISP pppd options file. If you want to authenticate using CHAP, add the same entry to the CHAP item instead.

```
# pppd: pap-secrets
# Secrets for authentication using PAP
# client        server  secret                  IP addresses
your_ISP_login    *     your_password
```

The "*" can be replaced with the IP address or name of the server you are dialling into if you know it. Usually, an asterisk is sufficient.

Important

If you do not know if your ISP is using PAP or CHAP authenfication just provide the information on both pap-secrets and chap-secrets files. They have exactly the same structure.

Backup the ppp.lrp package.

Step 4: configure your interfaces file

Trough the LEAF configuration menu type 1 to access to the network configuration menu and 1 again to edit your `/etc/network/interfaces` file. Enter the following information:

```
auto lo ppp0 eth0

iface lo inet loopback

iface ppp0 inet ppp
        provider provider

iface eth0 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        broadcast 192.168.1.255
```

The "auto" statement declares all the interfaces that will be automatically set up at boot time. This job will be carried out by the "ifup -a" statement in the `/etc/init.d/networking` script.

The syntax of "iface" statements is explained in the Bering-uclibc's installation guide.

Backup the etc.lrp package.

Step 5: configure Shorewall

Through the LEAF packages configuration menu, choose shorwall and check the two following files:

A) The `interfaces` file (entry 3) defines your interfaces. Here connection to the net goes through ppp0 and the connection to the internal network through eth0. So we must set:

```
(...)
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     ppp0            -
loc     eth0            detect          routestopped
```

```
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Warning

Do not forget the "-" under the BROADCAST heading for the net/ppp0 entry.

B) The `masq` file (entry 7). With a dial-up modem setup it should look like:

```
(...)
#INTERFACE              SUBNET
ppp0                    eth0
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Backup the shorwall.lrp package.

Step 6: Make the connection persistent (optional)

If you want to make your connection persistent, i.e. redial automatically your ISP when your line drops down, do the following:

Go back to the option 1) of the ppp configuration file menu to edit the `/etc/ppp/peers/provider` file and add the following options after the "baud rate" entry:

```
(...)
115200          # baud  rate
persist
holdoff 10
(...)
```

backup the ppp.lrp package.

Step 7: reboot...

Your modem connection should be established automatically. Type *plog* to check the login sequence with your ISP. If there is no output check the logs in `/var/log/` to get a clue on potential problems.

Tip

If you want to be sure that your modem and/or script parameters are OK before backing up ppp.lrp, you can launch the connection manually just by typing *pon*. Use the *plog* command to see how the connection is going and *poff* to close down your ppp connection.

ppp-filter.lrp

ppp-filter.lrp needs to be renamed to ppp.lrp before use and uses libpcap.lrp (which also needs to be loaded in `lrpkg.cfg/leaf.cfg`). The filter version can be used to specify a packet filter to be applied to data packets to determine which packets are to be regarded as link activity, and therefore reset the idle timer, or cause the link to be brought down in demand-dialing mode. The configuration (except for the filter-part) is done like the ppp package.

You can enable active filtering by setting (from the pppd man-page):

```
active-filter filter-expression
```

This option is useful in conjunction with the idle option if there are packets being sent or received regularly over the link (for example, routing information packets) which would otherwise prevent the link from ever appearing to be idle. The filter-expression syntax is as described for tcpdump, except that qualifiers which are inappropriate for a PPP link, such as ether and arp, are not permitted. Generally the filter expression should be enclosed in single-quotes to prevent whitespace in the expression from being interpreted by the shell.

Chapter 6. PPPoE configuration

Eric Wolzak <ericw at users.sourceforge.net>
Eric Spakman <espakman at users.sourceforge.net>

Revision History

Revision 0.4	2004-05-04	ES
Update for leaf.cfg		
Revision 0.3	2004-03-05	ES
Update for Bering-uClibc		
Revision 0.2	2002-04-14	EW
corrected and edited		
Revision 0.1	2002-03-15	EW
initial revision		

Objectives

We assume here that you want to connect your LEAF router to the Internet via an ADSL PPPoE connection. What is described here corresponds to section 3.2.3 of the DSL How-To [http://en.tldp.org/HOWTO/DSL-HOWTO/configure.html] document. Your ADSL modem is supposed to be connected to eth0, while the traffic to your internal network goes through eth1.

The PPP-Howto [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html] and the DSL-Howto [http://en.tldp.org/HOWTO/DSL-HOWTO/index.html] are two useful references for this section.

Comments on this section should be addressed to its maintainer: Eric Spakman <espakman at users.sourceforge.net>.

Step 1: Declare the ppp and pppoe packages

Those two packages are provided on the standard Bering-uClibc floppy disk, but are not activated by default.

Boot a Bering-uClibc floppy image. Once the LEAF menu appears get access to the linux shell by (q)uitting the menu. Edit the `lrpkg.cfg` (pre Bering-uClibc-2.2.0) or `leaf.cfg` (Bering-uClibc-2.2.0 onwards) file and REPLACE the dhcpcd entry by ppp,pppoe in the list of packages to be loaded at boot. Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-install.html] to learn how to do that.

Step 2: Declare the ppp and pppoe modules

In order to have a PPPoE connection working, you need to have ppp and pppoe support enabled through the appropriate kernel modules. You also need to declare the driver(s) module(s) of your network card(s). In the following example, we assume that both ethernet interfaces are provided through a standard ne 2000 PCI card.

All the modules which are necessary for a PPPoE connection are provided on the standard Bering-uClibc floppy. You just need to "declare" them since they are not loaded by default. As far as your network cards are concerned, the most popular driver modules are provided in `/lib/modules` but you might need to download the one corresponding to your own hardware from the Bering-uClibc modules download area [http://cvs.sourceforge.net/viewcvs.py/leaf/bin/bering-uclibc/packages/]. Refer to the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-lrpkg.html] to learn how to do that.

To declare your modules, go to the LEAF Packages configuration menu and choose modules. Enter 1) to edit the `/etc/modules` file and enter the following information:

```
# 8390 based ethernet cards
8390
ne2k-pci

# Modules needed for PPP/PPPOE connection
slhc
n_hdlc
ppp_generic
ppp_synctty
pppox
pppoe

# Masquerading 'helper' modules
ip_conntrack_ftp
ip_conntrack_irc
ip_nat_ftp
ip_nat_irc
```

Important

The `/etc/modules` file provided in the Bering-uClibc distro is already setup with those entries commented out. Just remove the leading # sign to activate the corresponding module.

Backup the modules.lrp package.

Step 3: Configure ppp

In the normal situation, you won't have to do anything here, the ppp is preconfigured for the standard situation.

Connection with your ISP will be handled by PPP. The PPP Howto [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html] document will give you very detailed information about this protocol and how to set-up its numerous parameters.

Please refer to the Serial Modem configuration [http://leaf.sourceforge.net/doc/guide/bucu-ppp.html] section of this user's guide to learn how to configure your ppp package.

The default options provided with the ppp.lrp should work and if you are not familiar with ppp leave them at first. After you get a connection you can "fine tune" your setup.

Step 4: Configure pppoe

Through the LEAF Package configuration menu choose pppoe. The following menu will appear:

```
                    pppoe configuration files

      1) DSL pppd options

q) quit
--------------------------------------------------------------------------------
      Selection:
```

Entry 1) allows you to adjust the parameter of your ppp connection through the `/`

`etc/ppp/peers/dsl-provider` file. The most important argument is the *user* parameter which defines your login name.

Replace the field following the user statement in the `/etc/ppp/peers/dsl-provider` ["login@isp.com"] by the login name provided by your ISP.

```
# Configuration file for PPP, using PPP over Ethernet
# to connect to a DSL provider.
#
plugin /usr/lib/pppd/rp-pppoe.so

# MUST CHANGE: Uncomment the following line, replacing the user@provider.net
# by the DSL user name given to your by your DSL provider.
# (There should be a matching entry in /etc/ppp/pap-secrets with the password.)
user "eric12345@foobar.com"

(...)
```

Through the LEAF packages configuration menu get access to ppp configuration. The following menu will show-up

```
                        ppp configuration files

       1) ISP pppd options
       2) ISP login script
       3) System wide pppd options
       4) chap secret
       5) pap secret

  q) quit
  ------------------------------------------------------------------------
       Selection:
```

Entry 5) allows you to edit the `/etc/ppp/pap-secrets`. Enter in this file the login and password provided by your ISP. Your login name must EXACTLY match the one given in the previous `/etc/ppp/peers/dsl-provider` file. If you have special characters in secret or username, you should put them in quotes

```
# This is a pap-secrets file
#
#papname * papsecret
"eric12345@foobar.com" * "secretfoo"
```

Backup both pppoe and ppp packages.

Step 5: Configure your interfaces file

Trough the LEAF configuration menu type 1 to access to the network configuration menu and 1 again to edit your `/etc/network/interfaces` file. Enter the following information:

```
auto lo ppp0 eth1

iface lo inet loopback

iface ppp0 inet ppp
```

```
        pre-up ip link set eth0 up
        provider dsl-provider eth0

iface eth1 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        broadcast 192.168.1.255
```

In this `/etc/network/interfaces` file the lo, ppp0 and eth1 interfaces are brought up automatically when the *ifup -a* statement is executed at boot time by the `/etc/init.d/networking` script.

The "iface ppp0 inet ppp" says:

- Execute the "ip link set eth0 up" command BEFORE ppp0 is activated (pre-up statement)
- Execute the `/sbin/pon dsl-provider eth0` script to establish the PPPoE connection. The `dsl-provider` file used as input by `/sbin/pon` is provided in the pppoe.lrp package.

The "iface eth1 inet static" defines the internal address of the router.

Backup the etc.lrp package.

Step 6: Configure Shorewall

Through the LEAF packages configuration menu, choose shorwall and check the three following files:

A) The `interfaces` file (entry 3) defines your interfaces. Here connection to the net goes through ppp0. So we must set:

```
(...)
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     ppp0            -         routefilter
loc     eth1            detect          routestopped
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Warning

Do not forget the "-" under the BROADCAST heading for the net/ppp0 entry.

B) The `masq` file (entry 7). With a dial-up modem setup it should look like:

```
(...)
#INTERFACE              SUBNET
ppp0                    eth1
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

C) You may also need to edit the `config` file (entry 12) to adjust the CLAMPMSS variable to "yes":

```
(...)
# Set this variable to "Yes" or "yes" if you want the TCP "Clamp MSS to PMTU"
# option. This option is most commonly required when your internet
# interface is some variant of PPP (PPTP or PPPoE). Your kernel must
#
```

```
# If left blank, or set to "No" or "no", the option is not enabled.
#
CLAMPMSS="yes"
(...)
```

Backup the shorwall.lrp package.

Step 7: Reboot...

Your modem connection should be established automatically. Type *plog* to check the login sequence with your ISP. If there is no output check the various logs in `/var/log/` to get a clue on potential problems.

An example: a PPPoE connection with a two PCMCIA cards setup

C. Hostelet is using an old laptop as a Bering-uClibc router. His hardware configuration consists of one HP Omnibook 3000 laptop (Pentium 233Mhz, 144MB Ram, CD-Rom drive module, no floppy, no HDD), one Xircom CEM56 Modem/ethernet PCMCIA card and one 3Com 3C589 PCMCIA card. The connection to the net is provided through the first PCMCIA card connected to an Alcatel SpeedTouch Home ethernet modem which gives him access to France Telecom "Netissimo" ADSL service. The connection to the local network is done trough the second PCMCIA card.

Here is his `/etc/network/interfaces` file:

```
auto lo

iface lo inet loopback

iface eth0 inet static
        address 10.0.0.1
        netmask 255.255.255.0
        broadcast 10.0.0.255
        up pon dsl-provider eth0
        up shorewall restart
        down shorewall stop
        down poff

iface eth1 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        broadcast 192.168.1.255
        up /etc/init.d/dnscache restart
        down /etc/init.d/dnscache stop
```

Only lo is brought up automatically at boot time. eth0 and eth1 are brought up by the PCMCIA cardmgr program which calls the `/etc/pcmcia/network` script.

The connection with the Alcatel speedtouch modem is done through the eth0 interface at address 10.0.0.1

Once the eth0 interface is up the pppd daemon is called by the pon script. Shorewall must then be restarted since eth0 was not available at boot time

Once the eth1 interface is up we restart dnscache which could not start at boot time since eth1 was not

available.

Chapter 7. PPTP/PPPoA configuration

Jacques Nilo <jnilo at users.sourceforge.net>
Eric Spakman <espakman at users.sourceforge.net>

Revision History

Revision 0.3	2004-03-06	ES
Update for Bering-uClibc		
Revision 0.2	2002-04-14	JN
initial revision		

Objectives

We assume here that you want to connect your LEAF router to the Internet via an Alcatel SpeedTouch home ADSL modem which supports both PPPoE and PPPoA connections. The PPPoE connection is covered in another section. For the PPPoA connection, we assume that your modem is connected to a dedicated NIC as eth0 and will communicate with your router through the pptp protocol. What is described here corresponds to section 3.2.5 of the DSL How-To [http://en.tldp.org/HOWTO/DSL-HOWTO/configure.html] document. The traffic to your internal network goes through eth0 while access to the Internet via PPPoA goes through ppp0.

The PPP-Howto [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html], the PPTP-Client [http://pptpclient.sourceforge.net] project and the DSL-Howto [http://en.tldp.org/HOWTO/DSL-HOWTO/index.html] are two useful references for this section.

Thanks to Eric de Thouars [http://www.xs4all.nl/~dorus/linux/] who suggested the required adjustment to Shorewall for this setup to work properly. Comments on this section should be addressed to its maintainer: Eric Spakman <espakman at users.sourceforge.net>.

Step 1: declare the ppp and the pptp packages

Boot a Bering floppy image. Once the LEAF menu appears get access to the linux shell by (q)uitting the menu. Edit the `lrpkg.cfg` file and REPLACE the dhcpcd entry by ppp,pptp in the list of packages to be loaded at boot. Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-install.html] to learn how to do that.

Your `lrpkg.cfg` file will then look like (adjust to your tastes):

```
root,etc,local,modules,iptables,ppp,pptp,keyboard,shorwall,ulogd,dnscache,weblet
```

Important

The line ("root ... dnscache,weblet") must be typed as a single one in `lrpkg.cfg`

The ppp package is provided on the standard Bering-uClibc floppy. The pptp.lrp package is available here [http://cvs.sourceforge.net/viewcvs.py/leaf/bin/bering-uclibc/packages/]. http://leaf.sourceforge.net/doc/guide/buci-install.html

Step 2: declare the ppp modules

In order to have a PPTP/PPPoA connection working, you need to have ppp support enabled through the

appropriate kernel modules. You also need to declare the driver(s) module(s) of your network card(s). In the following example, we assume that both ethernet interfaces are provided through a standard ne 2000 PCI card.

All the modules which are necessary for a PPTP/PPPoA connection are provided on the standard Bering floppy. You just need to "declare" them since they are not loaded by default. As far as your network cards are concerned, the most popular driver modules are provided in `/lib/modules` but you might need to download the one corresponding to your own hardware from the Bering modules CVS area [http://cvs.sourceforge.net/viewcvs.py/leaf/bin/bering-uclibc/packages/]. Refer to the Bering installation guide [http://leaf.sourceforge.net/doc/guide/buci-install.html] to learn how to do that.

To declare your modules, go to the LEAF Packages configuration menu and choose modules. Enter 1) to edit the `/etc/modules` file and enter the following information:

```
# 8390 based ethernet cards
8390
ne2k-pci

# Modules needed for PPTP/PPPoA connection
slhc
n_hdlc
ppp_generic
ppp_async

# Masquerading 'helper' modules
ip_conntrack_ftp
ip_conntrack_irc
ip_nat_ftp
ip_nat_irc
```

Important

The `/etc/modules` file provided in the Bering-uClibc distro is already setup with those entries commented out. Just remove the leading # sign to activate the corresponding module.

Backup the modules.lrp package.

Step 3: configure ppp

Connection with your ISP will be handled by PPP. The PPP Howto [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html] document will give you very detailed information about this protocol and how to set-up its numerous parameters.

Through the LEAF packages configuration menu get access to ppp configuration. The following menu will show-up

```
                        ppp configuration files

       1) ISP pppd options
       2) ISP login script
       3) System wide pppd options
       4) chap secret
       5) pap secret

  q) quit
  --------------------------------------------------------------------------
       Selection:
```

Enter 1) and 2) and empty out the corresponding files completely

Enter 3) allows you to adjust the parameter of your ppp connection through the `/etc/ppp/options` file. This file must contain:

```
debug
name "ISPUserID"
noauth
noipdefault
defaultroute
```

Edit either the CHAP (Entry 4) or PAP (Entry 5) option to set up how your system authenticates.

For PAP authentication, choose the PAP option and add a line saying "<ISPUserID> * <ISPUserPassword> to the bottom of the file. <ISPUserID> is the same entry that you made in Entry 3) - the "System wide pppd options" file. The <ISPUserPassword> entry is self-explanatory. The "*" can be replaced with the IP address or name of the server you are dialling into if you know it. Usually, an asterisk is sufficient.

If you want to authenticate using CHAP, add the same entry to the CHAP item instead.

Backup the ppp.lrp package.

Step 4: configure your interfaces file

Trough the LEAF configuration menu type 1 to access to the network configuration menu and 1 again to edit your `/etc/network/interfaces` file. Enter the following information:

```
auto lo eth0 eth1

iface lo inet loopback

iface eth0 inet static
        address 10.0.0.1
        netmask 255.255.255.0
        broadcast 10.0.0.255
        up pptp 10.0.0.138

iface eth1 inet static
        address 192.168.1.254
        masklen 255.255.255.0
        broadcast 192.168.1.255
```

In this `/etc/network/interfaces` file the lo, eth0 and eth1 interfaces are brought up automatically when the *ifup -a* statement is executed at boot time by the `/etc/init.d/networking` script.

The "iface eth0 inet static" section defines the external address of the router and says:

- Bring up eth0 at address 10.0.0.1
- Execute the `pptp 10.0.0.138` command once eth0 is up to establish the PPTP/PPPoA connection.

The "iface eth1 inet static" defines the internal address of the router.

Backup the etc.lrp package.

Step 5: configure Shorewall

Through the LEAF packages configuration menu, choose shorwall and check the three following files:

A) The `interfaces` file (entry 3) defines your interfaces. Here connection to the net goes through ppp0. So we must set:

```
(...)
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     ppp0            -         routefilter
adsl eth0         10.0.0.255
loc     eth1            detect          routestopped
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Warning

Do not forget the "-" under the BROADCAST heading for the net/ppp0 entry.

B) Add the following line to `/etc/shorewall/policy`. Now the policy for traffic between the firewall and the adsl zone is set to ACCEPT:

```
(...)
fw      adsl    ACCEPT
```

C) The `masq` file (entry 8). With a dial-up modem setup it should look like:

```
(...)
#INTERFACE              SUBNET
ppp0                    eth1
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

D) You may also need to edit the `config` file (entry 12) to adjust the CLAMPMSS variable to "yes":

```
(...)
# Set this variable to "Yes" or "yes" if you want the TCP "Clamp MSS to PMTU"
# option. This option is most commonly required when your internet
# interface is some variant of PPP (PPTP or PPPoE). Your kernel must
#
# If left blank, or set to "No" or "no", the option is not enabled.
#
CLAMPMSS="yes"
(...)
```

Backup the shorwall.lrp package.

Step 7: reboot...

Your modem connection should be established automatically. Type *plog* to check the login sequence with your ISP. If there is no output check the logs in `/var/log/` to get a clue on potential problems.

Chapter 8. PPPoA configuration

Jacques Nilo <jnilo at users.sourceforge.net>
Eric Spakman <espakman at users.sourceforge.net>

Revision History

Revision 0.3 Initial document	2001-05-20	JN
Revision 0.4 Update for Bering-uClibc	2004-03-05	ES
Revision 0.5 Update for leaf.cfg	2004-05-04	ES

Objectives

We assume here that you want to connect your LEAF router to the Internet via PPPoA. The PPPoE connection [http://leaf.sourceforge.net/doc/guide/bucu-pppoe.html] is covered in another section of this user's guide. So is the PPTP/PPPoA connection [http://leaf.sourceforge.net/doc/guide/bucu-pppoa.html]. What is described here corresponds to section 3.2.4 of the DSL How-To [http://en.tldp.org/HOWTO/DSL-HOWTO/configure.html] document. The traffic to your internal network goes through eth0 while access to the Internet via PPPoA goes through ppp0.

The PPP-Howto [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html] and the DSL-Howto [http://en.tldp.org/HOWTO/DSL-HOWTO/index.html] are two useful references for this section.

Comments on this section should be addressed to its maintainer: Eric Spakman <espakman at users.sourceforge.net>.

Step 1: declare the pppoatm package

In order to be able to get connected through PPPoA you will the pppoatm.lrp and libatm.lrp packages together with ppp.lrp.

Boot your Bering-uClibc floppy image. Once the LEAF menu appears get access to the linux shell by (q)uitting the menu. Edit the `lrpkg.cfg` (pre Bering-uClibc-2.2.0) or `leaf.cfg` (Bering-uClibc-2.2.0 onwards) file and REPLACE the dhcpcd entry by pppoatm and libatm in the list of packages to be loaded at boot. Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-lrpkg.html] to learn how to do that.

The pppoatm.lrp and libatm.lrp packages are available here [http://cvs.sourceforge.net/viewcvs.py/leaf/bin/packages/uclibc-0.9/20/].

Step 2: declare the ppp and pppoatm modules

In order to have a PPPoA connection working, you need to have both ppp and pppoatm support enabled through the appropriate kernel modules. You also need to declare the driver(s) module(s) of your network card(s). In the following example, we assume that the external connection to the Internet is provided by a Madge Ambassador ATM/PCI card while the internal network goes through a standard ne 2000 PCI card.

All the modules which are necessary for ppp support are provided on the standard Bering floppy. You just need to "declare" them since they are not loaded by default. As far as the pppoatm module is concerned you will have to download it from the Bering-uClibc modules CVS area

[http://cvs.sourceforge.net/viewcvs.py/leaf/bin/bering-uclibc/packages/] and store it in `/lib/modules`.

ATM drivers are available here [http://cvs.sourceforge.net/viewcvs.py/leaf/bin/bering-uclibc/packages/].

To declare your modules, go to the LEAF Packages configuration menu and choose modules. Enter 1) to edit the `/etc/modules` file and enter the following information:

```
# 8390 based ethernet cards
8390
ne2k-pci

# Modules needed for PPP connection
slhc
ppp_generic

# PPPOA support
pppoatm

# ATM-PCI "st" drivers
ambassador

# Masquerading 'helper' modules
ip_conntrack_ftp
ip_conntrack_irc
ip_nat_ftp
ip_nat_irc
```

Backup the modules.lrp package.

Step 3: configure pppoatm

Connection with your ISP will be handled by PPP. The PPP Howto [http://en.tldp.org/HOWTO/PPP-HOWTO/index.html] document will give you very detailed information about this protocol and how to set-up its numerous parameters.

Through the LEAF packages configuration menu get access to pppatm configuration. The following menu will show-up:

```
                        pppoatm configuration files

       1) DSL pppd options

  q) quit
  ------------------------------------------------------------------------------
       Selection:
```

Enter 1) and adjust the corresponding `/etc/ppp/peers/dsl-provider` file:

```
#
# Adjust here VP/VC - depends on country & ISP
# UK/BT: 0.38 - US/BE/FR: 8.35
#
plugin /usr/lib/pppd/pppoatm.so 0.38
#
# If chap or pap identification uncomment the #name "ISPUserID" line
# and replace ISPUserID with your ISP user name
```

```
# There should be a matching entry in /etc/ppp/pap-secrets or chap-secrets
#
#name "ISPUserID"
lock
noipdefault
noauth
defaultroute
hide-password
lcp-echo-interval 20
lcp-echo-failure 3
maxfail 0
persist
```

The most important parameters in this file are the VP.VC combination which depends on your country and/or your ISP and the name parameter.

Through the LEAF packages configuration menu get access to ppp configuration. The following menu will show-up

```
                        ppp configuration files

       1) ISP pppd options
       2) ISP login script
       3) System wide pppd options
       4) chap secret
       5) pap secret

  q) quit
  ------------------------------------------------------------------------------
       Selection:
```

Edit either the CHAP (Entry 4) or PAP (Entry 5) option to set up how your system authenticates. If you edit chap, replace #ISPUserID and ISPUserPassword with the relevant information.

```
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
#ISPUserID      *       ISPUserPassword
```

ISPUserID must exactly match the entry that you made for the name parameter in pppoatm Entry 1) "DSL pppd options" file. The "*" can be replaced with the IP address or name of the server you are dialling into if you know it. Usually, an asterisk is sufficient.

If you want to authenticate using PAP, add the same entry to the PAP item instead.

Backup the pppoatm.lrp and ppp.lrp packages.

Step 4: configure your interfaces file

Trough the LEAF configuration menu type 1 to access to the network configuration menu and 1 again to edit your `/etc/network/interfaces` file. Enter the following information:

```
auto lo ppp0 eth0

iface lo inet loopback
```

```
iface ppp0 inet ppp
        provider dsl-provider

iface eth0 inet static
        address 192.168.1.254
        netmask 255.255.255.0
        broadcast 192.168.1.255
```

In this `/etc/network/interfaces` file the lo, ppp0 and eth0 interfaces are brought up automatically when the *ifup -a* statement is executed at boot time by the `/etc/init.d/networking` script.

The "iface ppp0 inet ppp" section defines the external address of the router and activates the `pon` script

The "iface eth0 inet static" defines the internal address of the router.

Backup the etc.lrp package.

Step 5: configure Shorewall

Through the LEAF packages configuration menu, choose shorwall and check the three following files:

A) The `interfaces` file (entry 3) defines your interfaces. Here connection to the net goes through ppp0. So we must set:

```
(...)
#ZONE   INTERFACE       BROADCAST       OPTIONS
net     ppp0            -
loc     eth0            detect          routestopped
#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

Warning

Do not forget the "-" under the BROADCAST heading for the net/ppp0 entry.

B) The `masq` file (entry 8). It should look like:

```
(...)
#INTERFACE              SUBNET
ppp0                    eth0
#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

Backup the shorwall.lrp package.

Step 7: reboot...

Your PPPoA connection should be established automatically. Type *plog* to check the login sequence with your ISP. If there is no output check the various logs in `/var/log/` to get a clue on potential problems.

Chapter 9. ez-ipupdate configuration

Jacques Nilo <jnilo at users.sourceforge.net>
K.-P. Kirchdörfer <kapeka at users.sourceforge.net>

Revision History

Revision 0.1	2001-05-20	JN
Initial document		
Revision 0.2	2004-02-11	kp
Update for Bering-uClibc		

About ez-ipupdate

What is ez-ipupdate?

Ez-ipupdate is a small utility for updating your host name IP for any of the dynamic DNS service offered at:

- http://www.ez-ip.net
- http://www.justlinux.com
- http://www.dhs.org
- http://www.dyndns.org
- http://www.ods.org [http://www.ods.org]
- http://gnudip.cheapnet.net [http://gnudip.cheapnet.net] (GNUDip)
- http://www.dyn.ca (GNUDip)
- http://www.tzo.com
- http://www.easydns.com
- http://www.dyns.cx
- http://www.hn.org
- http://www.zoneedit.com

This package has been developed & is supported by Angus Mackay [http://gusnet.cx/proj/ez-ipupdate].

The key features are: support for multiple service types and updating your IP address if it changes.

Feedback

Comment on the LEAF package can be sent to the authors.

Declare the ezipupd.lrp package

Download the ezipupd.lrp [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/leaf/bin/packages/uclibc-0.9/20/ezipupd.lrp?rev=HEAD&content-type=application/octet-stream] package and copy the package to your Bering-uClibc diskette.

Boot a Bering-uClibc floppy image. Once the LEAF menu appears get access to the linux shell by (q)uitting the menu. Edit the `lrpkg.cfg` (pre Bering-uClibc-2.2.0) or `leaf.cfg` (Bering-uClibc-2.2.0 onwards) file and add `ezipupd.lrp` in the list of packages to be loaded at boot. Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-install.html] to learn how to do that.

Configuring ez-ipupdate

You can edit the ez-ipupdate configuration file through the package configuration menu:

```
                    ez-ipupd configuration files

      1) configuration file
      2) startup script

q) quit
--------------------------------------------------------------------------------
      Selection:
```

The parameters allowed in the configuration file are the followings:

```
address         usage: address=[ip address]
cache-file         usage: cache-file=[cache file]
cloak-title         usage: cloak-title=[title]
daemon          usage: daemon=[command]
execute          usage: execute=[shell command]
debug                 usage: debug
foreground         usage: foreground
pid-file          usage: pid-file=[file]
host                 usage: host=[host]
interface          usage: interface=[interface]
mx                  usage: mx=[mail exchanger]
max-interval          usage: max-interval=[number of seconds between updates]
notify-email          usage: notify-email=[address to email if bad things happen]
offline          usage: offline
retrys          usage: retrys=[number of trys]
server                  usage: server=[server name]
service-type          usage: service-type=[service type]
timeout          usage: timeout=[sec.millisec]
resolv-period          usage: resolv-period=[time between failed resolve attempts
period          usage: period=[time between update attempts]
url                usage: url=[url]
user                usage: user=[user name][:password]
run-as-user         usage: run-as-user=[user]
run-as-euser          usage: run-as-euser=[user] (this is not secure)
wildcard          usage: wildcard
quiet                usage: quiet
connection-type         usage: connection-type=[connection type]
request                usage: request=[request uri]
partner                usage: partner=[easydns partner]
```

Here is how it could look like:

```
service-type=zoneedit
user=myname:mypassword
interface=eth0
host=mydomain.com
#notify-email=john.doe@mydomain.com
# other options:
#address=<ip address>
#cache-file=/tmp/ez-ipup
#daemon
#debug
#foreground
#host=<host>
#interface=<interface>
#mx=<mail exchanger>
#retrys=<number of trys>
#run-as-user=<user>
#run-as-euser=<user>
#server=<server name>
#timeout=<sec.millisec>
#max-interval=<time in seconds>
#notify-email=<email address>
#period=<time between update attempts>
#url=<url>
```

The four most important entries for a typical LEAF Bering-uClibc installation will be explained below:

- service-type - make shure to add the according service-type. See above the list of available services.
- user - here you have to provide your username and password for the choosen service seperated by a colon.
- interface - this defines your interface to the internet - the one which is changing from time to time and whose ip should be changed at your dynamic DNS service (usually eth0 or ppp0)
- host - the host(s) you like to have been updated. It is possible to update more than one host for a service-type. To allow that, add all your hosts separated by a comma.

Please note: The options `cache-file` and `daemon` aren't needed. `notify-email` doesn't work today.

You can also run ez-ipupdate in interractive mode. The commands are:

```
null ezip pgpow dhs dyndns dyndns-static dyndns-custom ods
 tzo easydns easydns-partner gnudip justlinux dyns hn zoneedit
 heipv6tb
                   usage: ez-ipupdate [options]

Options are:
 -a, --address <ip address>    string to send as your ip address
 -b, --cache-file <file>       file to use for caching the ipaddress
 -c, --config-file <file>      configuration file, almost all arguments can be
                               given with: <name>[=<value>]
                               to see a list of possible config commands
                               try "echo help | ez-ipupdate -c -"
 -d, --daemon                  run as a daemon periodicly updating if
                               necessary
 -e, --execute <command>       shell command to execute after a successful
```

```
                                update
  -f, --foreground              when running as a daemon run in the foreground
  -F, --pidfile <file>          use <file> as a pid file
  -g, --request-uri <uri>       URI to send updates to
  -h, --host <host>             string to send as host parameter
  -i, --interface <iface>       which interface to use
  -L, --cloak_title <host>      some stupid thing for DHS only
  -m, --mx <mail exchange>      string to send as your mail exchange
  -M, --max-interval <# of sec> max time in between updates
  -N, --notify-email <email>    address to send mail to if bad things happen
  -o, --offline                 set to off line mode
  -p, --resolv-period <sec>     period to check IP if it can't be resolved
  -P, --period <# of sec>       period to check IP in daemon
                                mode (default: 1800 seconds)
  -q, --quiet                   be quiet
  -r, --retrys <num>            number of trys (default: 1)
  -R, --run-as-user <user>      change to <user> for running, be ware
                                that this can cause problems with handeling
                                SIGHUP properly if that user can't read the
                                config file. also it can't write it's pid file
                                to a root directory
  -Q, --run-as-euser <user>     change to effective <user> for running,
                                this is NOT secure but it does solve the
                                problems with run-as-user and config files and
                                pid files.
  -s, --server <server[:port]>  the server to connect to
  -S, --service-type <server>   the type of service that you are using
                                try one of: null ezip pgpow dhs
                                dyndns dyndns-static dyndns-custom
                                ods tzo easydns easydns-partner
                                gnudip justlinux dyns hn zoneedit
                                heipv6tb
  -t, --timeout <sec.millisec>  the amount of time to wait on I/O
  -T, --connection-type <num>   number sent to TZO as your connection
                                type (default: 1)
  -U, --url <url>               string to send as the url parameter
  -u, --user <user[:passwd]>    user ID and password, if either is left blank
                                they will be prompted for
  -w, --wildcard                set your domain to have a wildcard alias
  -z, --partner <partner>       specify easyDNS partner (for easydns-partner
                                services)
      --help                    display this help and exit
      --version                 output version information and exit
      --credits                 print the credits and exit
      --signalhelp              print help about signals
```

Using ez-ipupdate

Through dhclient exit-hook script

```
reload_all() {
  /sbin/shorewall restart
echo "Starting ez-ipupd from dhclient ..."
  /etc/init.d/ez-ipupd start

}
```

Through ppp /etc/ppp/ip-up script

All you need is to add the command **/etc/init.d/ez-ipupd -start** to `/etc/ppp/ip-up.`

```
# Main Script starts here
#
/etc/init.d/ez-ipupd start

run-parts /etc/ppp/ip-up.d
[ -x /bin/beep ] && /bin/beep -f 600 -n -f 900 -n -f1200 -n -f1800
# last line
```

Chapter 10. Configuring IPv6

Eric de Thouars <dorus at users.sourceforge.net>

Revision History

Revision 0.1	2003-08-11	ET
Initial document		
Revision 0.2	2003-08-13	ET
Links to IPv6 packages and 6wall documentation corrected		
Revision 0.3	2003-08-29	ET
Added description for OpenSSH daemon		

Introduction

IPv6 support in Bering-uClibc

Since version 2.0 of Bering-uClibc IPv6 is an officially supported feature. In previous versions of Bering-uClibc and in "plain" Bering very limited IPv6 functionality was available using the `ipv6.o` kernel module and the `ip` command, but no IPv6 applications were provided.

The IPv6 support of Bering-uClibc consists of

- a modules package with all necessary IPv6 kernel modules
- applications compiled with IPv6 enabled (if applicable)
- 6wall, an IPv6 firewall based on Shorewall

What can be found in this document

This chapter consists of two parts. In the next section the IPv6 configuration of Bering-uClibc is described. The rest of the sections contain application specific notes regarding IPv6.

IPv6 configuration

Objectives

These instructions are for those who want to use their Bering-uClibc system not only as an IPv4 router/firewall but also as an IPv6 router/firewall. This document assumes that you already have a some knowledge about Bering-uClibc.

A good start for more information on IPv6 is the Linux IPv6 HOWTO [http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/] and Peter Bieringer's IPv6 & Linux - HowTo [http://www.bieringer.de/linux/IPv6/IPv6-HOWTO/IPv6-HOWTO.html].

Prerequisites

Disk space

Depending on the other packages and modules that you have installed on your system, one floppy may

offer you enough disk space to put it all together. Check the Bering user's guide section about "Booting Bering from different boot-media [http://leaf.sourceforge.net/doc/guide/bubooting.html]" for tips on e.g. a two-floppy setup or other solutions.

Static IPv4 address

You can use your Bering-uClibc IPv6 router for stand-alone networks without additional requirements. However, if you want to connect to the Internet using IPv6 you need an IPv6 link to the Internet. Most of us currently don't have a native IPv6 connection to the Internet, but you can get access via an IPv6 tunnelbroker. In that case you establish an IPv4 tunnel with the tunnelbroker. The IPv6 traffic is sent through this tunnel from your network to the tunnelbroker and vice-versa. To setup this tunnel most tunnelbrokers require that you have a static IPv4 address assigned to you by your ISP.

Examples of tunnelbrokers are Freenet6 [http://www.freenet6.net], XS26 [http://www.xs26.net], SixXS [http://www.sixxs.net] and Hurricane Electric [http://ipv6tb.he.net].

Freenet6 uses the Tunnel Setup Protocol (TSP) to establish a tunnel between your IPv6 tunnel endpoint and their endpoint. A Bering-uClibc package with the TSP client is available. For more information check the chapter on freenet6.lrp.

uClibc libraries

The packages supporting IPv6 for Bering-uClibc are compiled against uClibc. You must use version 2.0 or later of Bering-uClibc to use these packages.

Step 1: Declare the ipv6 module

In order to have IPv6 working, you need to have IPv6 support enabled through the appropriate kernel module: `kernel/net/ipv6/ipv6.o`. There are two ways to do this:

- Use the `modules_ipv6.lrp` package
- Manually add the module to your existing `modules.lrp` package

Use `modules_ipv6.lrp`

if you are starting with fresh installation of Bering-uClibc or if you haven't heavily modified the modules package of you're system, this is probably the easiest approach. Rename this package to `modules.lrp` and replace the original package with this one. if you need more information on how to add/replace modules on your system, check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buc-install.html].

Note

Now reboot your system.

Manually add ipv6 module

The kernel module for IPv6 can be found in the kernel module tarball. This tarball can be downloaded from the Sourceforge FRS [http://sourceforge.net/project/showfiles.php?group_id=13751&package_id=67534]. Information on how to add a kernel module to your system can be found in the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buc-install.html].

Note

The size of this module is about 90Kb after compression in the package. Make sure that you have that much free space on the disk with the `modules.lrp` package.

To configure your module, go to the LEAF Packages configuration menu and choose modules. Enter 1) to edit the `/etc/modules` file and enter the following information:

```
(...)
# IPv6 support
ipv6
(...)
```

Note

Backup the `modules.lrp` package and reboot your system.

Check

After installing `modules_ipv6.lrp` or manually adding `ipv6.o` you can check if the module works by giving the following command:

```
Bering-uClibc -root-
# ip addr
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
    inet6 ::1/128 scope host
2: dummy0: <BROADCAST,NOARP> mtu 1500 qdisc noop
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
3: eth0: <BROADCAST,MULTICAST,UP>> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:40:95:1a:14:f4 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.120/24 brd 10.0.0.255 scope global eth0
    inet6 fe80::240:95ff:fe1a:14f4/10 scope link
4: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:40:95:1a:14:70 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.254/24 brd 192.168.1.255 scope global eth1
    inet6 fe80::240:95ff:fe1a:1470/10 scope link
5: sit0@NONE: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
```

Step 2: Declare the ipv6 packages

Copy the the following packages to one of your floppies:

- `radvd.lrp`
- `ip6table.lrp`
- `6wall.lrp`

These packages are provided in the IPv6 drop-in tarball which can be downloaded from the Sourceforge FRS [http://sourceforge.net/project/showfiles.php?group_id=13751&package_id=67534].

Detailed information on how to add packages to your system can be found in the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buc-install.html].

Modify the `lrpkg.cfg` file to load the new packages.

```
root,etc,local,modules,ppp,dnscache,weblet,iptables,shorwall,radvd,ip6table,6wall
```

Step 3: Configure IPv6 addresses

If all worked well, you should have seen some IPv6 addresses (inet6) appear on your interfaces in previous step. All these addresses have scope "local", this means that these are link-local addresses and can only be used on the network segment to which the interface is connected. But since you're working with a Bering-uClibc router it's highly unlikely that your network consists only of one segment, but rather of two or more.

If you want hosts on different segments to communicate with each other using IPv6, you need to assign these hosts either site-local or global addresses. global addresses need to be assigned to you by an ISP and site-local addresses are your to use freely within your own network (like the RFC1918 addresses for IPv4). Therefore, we'll start using site-local addresses first. If you have global addresses and don't want to use site-local addresses, you can skip this section and go directly to Step 6: Configure a 6to4 tunnel.

The prefixes used in this example are:

- eth0 - fec0:1::/64
- eth1 - fec0:2::/64

Invoke **lrcfg** and choose 1) Network configuration and then 1) Network Interfaces. In the interface file add an IPv6 address for your each of your interfaces that corresponds with the prefix for the interface.

```
(...)
iface eth0 inet6 static
        address fec0:1::1
        masklen 64

iface eth1 inet6 static
        address fec0:2::1
        masklen 64
(...)
```

Next, within **lrcfg** choose 2) Network options file. In this config file IPv6 forwarding can be enabled, which is nice for a router `;-)`.

```
(...)
ipv6_forward=yes
(...)
```

The effect of this configuration item is that on start-up the command **echo 1 > /proc/sys/net/ipv6/conf/all/forwarding** is given.

Note

Backup the `etc.lrp` package.

Step 4: Configure the Router Advertisement daemon

One of the features of IPv6 is the router advertisement mechanism. When a router advertises the network prefix to be used on a network segment, hosts on that segment can use the advertised prefix to automagically configure an IPv6 address. The router advertisement messages are also used by the hosts to configure the gateway address.

The `radvd.lrp` package contains a router advertisement daemon for Bering-uClibc. The configuration of the daemon is very straight forward. If in our example we want to use router advertisement on eth1, edit the `/etc/radvd.conf` file as follows:

```
interface eth1
{
        AdvSendAdvert on;
        prefix fec0:2::/64
        {
                AdvOnLink on;
                AdvAutonomous on;
        };
};
```

Note

Backup the `radvd.lrp` package and reboot the system.

Step 5: Check if the router is working properly

Check the ip addresses and the routing table with the following commands:

```
Bering-uClibc -root-
# ip -6 addr
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    inet6 ::1/128 scope host
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet6 fec0:1::1/64 scope site
    inet6 fe80::240:95ff:fe1a:14f4/10 scope link
4: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    inet6 fec0:2::1/64 scope site
    inet6 fe80::240:95ff:fe1a:1470/10 scope link

Bering-uClibc -root-
# ip -6 route
fe80::/10 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
fe80::/10 dev eth1  proto kernel  metric 256  mtu 1500 advmss 1440
fec0:1::/64 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
fec0:2::/64 dev eth1  proto kernel  metric 256  mtu 1500 advmss 1440
ff00::/8 dev eth0  proto kernel  metric 256  mtu 1500 advmss 1440
ff00::/8 dev eth1  proto kernel  metric 256  mtu 1500 advmss 1440
unreachable default dev lo  metric -1  error -101

Bering-uClibc -root-
```

```
# cat /proc/sys/net/ipv6/conf/all/forwarding
1
```

Now an IPv6 capable system (how to configure IPv6 on Win XP [http://www.microsoft.com/windowsxp/pro/techinfo/administration/ipv6/default.asp]) in one segment of your network should now be able to ping6 another IPv6 system in another segment connected to the router. Both should also be able to ping6 the router.

Ping6 is the IPv6 equivalent of ping and is provided by the `initrd.lrp` package.

Step 6: Configure a 6to4 tunnel

In the most luxurious case you have a native IPv6 connection to the internet. In that case you can follow Step 4 and 5 and substitute the site-local addresses used with your global addresses. if you're not that lucky, IPv6 access to the Internet can be achieved via a tunnelbroker (see Section - Prerequisites). You will get the necessary global addresses and prefix(es) from the tunnelbroker. This is what will be described below.

When connecting via a tunnelbroker, an IPv6-to-IPv4 (6to4) tunnel is established between your gateway and the tunnelbroker. To setup this tunnel you need the following information (imaginary information is given for the example):

- IPv4 address for the tunnel end-point of the tunnelbroker - `202.143.23.6`
- IPv6 address of the tunnelbroker - `3ffe:8280:0:2001::1`
- IPv6 address assigned to you - `3ffe:8280:0:2001::2`
- IPv6 prefix assigned to you (for use on your network) - `3ffe:8280:10:8560::/60`

Edit `/etc/network/interfaces` as follows:

```
(...)
#
# Definition of the Ipv6-to-IPv4 tunnel interface
#
#auto tun6to4
iface tun6to4 inet6 v4tunnel
        address 3ffe:8280:0:2001::2
        netmask 64
        gateway 3ffe:8280:0:2001::1
        endpoint 202.143.23.6
        ttl 254
(...)
```

Note

Backup the `etc.lrp` package.

Note

At this stage the `auto tun6to4` statement is commented-out. This is because no IPv6 firewall is active yet, so the tunnel is best brought up only when necessary and as soon as possible be brought down again. This can be done with the commands **ifup tun6to4** and **ifdown tun6to4**

Note

The gateway statement in the tunnel definition causes a default route to be created. However, it turns out that this is not working (maybe a bug in Linux IPv6 support). To solve this problem a route to the global address space (2000::/3) is added explicitly.

Step 7: Configure Shorewall

You need to allow the 6to4 tunnel traffic from the Bering-uClibc Firewall to the tunnelbroker. This traffic uses IP protocol 41.

With Shorewall 1.4.3 and later you can add a 6to4 tunnel definition in the `/etc/shorewall/tunnels` file. In the examples below it is assumed that your firewall zone is called "`fw`" and the Internet zone "`net`".

```
#TYPE    ZONE   GATEWAY         GATEWAY ZONE
(...)
6to4     net    202.143.23.6
(...)
```

More information on Shorewall and 6to4 tunnels can be found in the Shorewall documentation [http://www.shorewall.net/6to4.htm].

For versions older than 1.4.3 you can add some rules in the `/etc/shorewall/rules` file.

```
#ACTION  SOURCE             DEST              PROTO   DEST    SOURCE     ORIGINAL
#                                                    PORT    PORT(S)    DEST
(...)
#
# Accept 6to4 tunnel traffic from the firewall to tunnelbroker
#
ACCEPT   fw                 net:202.143.23.6  41
ACCEPT   net:202.143.23.6   fw                41
(...)
```

Note

Backup the `shorwall.lrp` package.

After restarting Shorewall, you should be able to ping6 or traceroute6 to IPv6 accessible hosts on the Internet. Some hosts that you could try are:

- www.ipv6.surfnet.nl
- www.linux-ipv6.org
- www.kame.net

Step 8: Configure the local network

In the following it is assumed that your local network is connected to eth1 and that the connection to the internet (over which the 6to4 tunnel will be established) goes via eth0.

Based on the information from your tunnelbroker, select the prefix of /64 to be used on the network segment connected to eth1. Then edit `/etc/network/interfaces` as follows:

```
(...)
iface eth1 inet6 static
        address fec0:2::1
        masklen 64
        up ip addr add 3ffe:8280:10:8560::1/64 dev eth1
(...)
```

Note

To define a second IPv6 address on an interface, don't use a second "iface" statement. This is not supported by ifupdown, use the "up" statement within the existing iface statement instead.

To advertise the selected global prefix as well as the site-local prefix from Step 4, edit the `/etc/radvd.conf` file as follows:

```
interface eth1
{
        AdvSendAdvert on;
        prefix fec0:2::/64
        {
                AdvOnLink on;
                AdvAutonomous on;
        };
        prefix 3ffe:8280:10:8560::/64
        {
                AdvOnLink on;
                AdvAutonomous on;
        };
};
```

Note

Backup the `etc.lrp` and the `radvd.lrp` packages.

After restarting the Router Advertisement daemon, any systems running on your local network that support autconfiguration for IPv6, they should be getting a global IPv6 address. If not, maybe the system needs to be rebooted first.

When a system on the local network has configured a global IPv6 address, it should now also be able to ping6 and traceroute6 to the hosts mentioned in Step 7.

Note

Keep in mind that at this moment there is still no IPv6 firewall active. This will be done in the

next step. Everybody has free access to all your systems with global addresses on the local network. So, only have the 6to4 tunnel up when you are doing some tests. After testing, bring down the tunnel as soon as possible.

Step 9: Configure 6wall, the IPv6 firewall

6wall is for IPv6 what Shorewall is for IPv4.

6wall is an IPv6 firewall which is derived from Shorewall version 1.4. So if you're familiar with Shorewall you should have no problem configuring 6wall. The best way to start is to read-up on Tom Eastep's excellent Shorewall documentation [http://www.shorewall.net/Documentation.htm]. After that, check the 6wall documentation [http://leaf-project.org/doc/howto/6wall.html] where specific issues for 6wall are described.

When you're done reading, you can configure 6wall. Go to the packages menu and select 6wall. The following menu will appear:

```
                        6wall configuration files

      1) Zones     Partition the network into Zones
      2) Ifaces    6wall Networking Interfaces
      3) Hosts     Define specific zones
      4) Policy    Firewall high-level policy
      5) Rules     Exceptions to policy
      6) Maclist   MAC verification
      7) Config    6wall Global Parameters
      8) Blacklist Blacklisted hosts
      9) SiteLocal Defines 'nositelocal' interface option
      10) Common    Common rules
      11) Init      Commands executed before [re]start
      12) Start     Commands executed after [re]start
      13) Stop      Commands executed before stop
      14) Stopped   Commands executed after stop

  q) quit
  ------------------------------------------------------------------------------
        Selection:
```

The configuration files that can be edited via the menu above are located in the `/etc/6wall/` directory. Below is the default configuration of some of these files.

The zones for IPv6 are described in `zones6`:

```
#ZONE   DISPLAY         COMMENTS
net     Net             Internet
loc     Local           Local networks
```

The interfaces for IPv6 are described in `interfaces6`:

```
#ZONE    INTERFACE      OPTIONS
net      tun6to4        nositelocal
loc      eth1
```

The policies for IPv6 are described in `policy6`:

```
#SOURCE         DEST            POLICY          LOG LEVEL
loc             net             ACCEPT
net             all             DROP              info
all             all             DROP              info
```

The rules for IPv6 are described in `rules6`:

```
#ACTION  SOURCE         DEST            PROTO   DEST    SOURCE     ORIGINAL
#                                               PORT    PORT(S)    DEST
#
#       Allow ping6 from the firewall
#
ACCEPT   fw             all             icmpv6  echo-request
#
#
#       Allow ping6 from the local network to the firewall
#
ACCEPT   loc            fw              icmpv6  echo-request
```

This configuration should get you started and you can modify these or other configuration files to suit your needs.

Note

Backup the `modules.lrp`, `6wall.lrp` and `etc.lrp` packages.

Now reboot your system and enjoy safe surfing on the IPv6 Internet !!

Tips and tricks

To be provided.....

IPv6 (enabled) applications

Overview

A number of applications are IPv6 specific while others are generic but with IPv6 support enabled. below an overview of the IPv6 (enabled) applications in Bering-uClibc is given. The following sections will go into the IPv6 specifics of these applications.

IPv6 applications

- **ping6** - provided by `busybox` in `initrd.lrp`
- **radvd** - provided by `radvd.lrp`
- **ip6tables** - provided by `ip6table.lrp`

- **6wall** - provided by `6wall.lrp`

IPv6 enabled applications

- **netstat** - provided by `busybox` in `initrd.lrp`
- **dnscache** - provided by `dnscache.lrp`
- **tinydns** - provided by `tinydns.lrp`
- **inetd** - provided by `root.lrp`
- **pppd** - provided by `pppd.lrp`
- **snmpd** - provided by `netsnmpd.lrp`
- **sshd** - provided by `sshd.lrp`, `libz.lrp` and `libcrpto.lrp`

ping6 & netstat

Overview

Description	ping6 and netstat from BusyBox
Source	www.busybox.net [http://www.busybox.net]
Version	1.0
Package	`initrd.lrp`

Configuration

No specific configuration for these applications is necessary.

Limitations & known problems

The known limitations and problems with these applications are listed below. If you happen to have a solution for these issues, please let us know.

- None.

radvd

Overview

Description	Router advertisement daemon
Source	v6web.litech.org/radvd [http://v6web.litech.org/radvd]
Version	0.7.2

Package	radvd.lrp

Configuration

Check the section on IPv6 configuration for more info on how to configure the routing advertisement daemon.

Limitations & known problems

The known limitations and problems with this application are listed below. If you happen to have a solution for these issues, please let us know.

- None.

ip6tables

Overview

Description	Netfilter application for IPv6
Source	www.netfilter.org [http://www.netfilter.org]
Version	1.2.8
Package	ip6table.lrp

Configuration

No explicit configuration is needed for ip6tables within Bering-uClibc since the complete netfilter configuration is done by 6wall.

Limitations & known problems

The known limitations and problems with this application are listed below. If you happen to have a solution for these issues, please let us know.

- To be provided.

6wall

Overview

Description	IPv6 firewall scripts for ip6tables
Source	LEAF CVS [http://leaf.sourceforge.net/devel/dorus]
Version	1.0.2
Package	6wall.lrp

Configuration

Check the section on IPv6 configuration for more info on how to configure the routing advertisement daemon.

Limitations & known problems

The known limitations and problems with this application are listed below. If you happen to have a solution for these issues, please let us know.

- See section "Limitations" in the 6wall documentation [http://leaf-project.org/doc/howto/6wall.html#sixwall1].

dnscache & tinydns

Overview

Description	Dns cache and dns server applications from Tinydns
Source	tinydns.org [http://tinydns.org] and the IPv6 patch from www.fefe.de/dns/ [http://www.fefe.de/dns/]
Version	1.0.5
Package	`dnscache.lrp` and `tinydns.lrp`

Configuration

The current version of the IPv6 patch adds support for AAAA records (those are the DNS records that store IPv6 numbers) and IPv6 addresses in PTR records. It also supports automatic internal lookup of some reserved IPv6 addresses (like "::1").

IPv6 related configuration is only applicable for **tinydns**. The AAAA records are configured in the private DNS server data file (`/etc/tinydns-private/root/data`) and/or the public DNS server data file (`/etc/tinydns-public/root/data`). See below for a sample configuration of the private DNS server data file with IPv6 addresses. The keyword "**6**" is used to define the IPv6 AAAA and PTR records. If you don't want the PTR record but only the AAAA record, use the keyword "**3**" instead.

```
.local.net::ns.local.net
.1.168.192.in-addr.arpa::ns.local.net
+ns.local.net:127.0.0.1
=hosta.local.net:192.168.1.1
=hostb.local.net:192.168.1.2
=firewall.local.net:192.168.1.254
6hosta.local.net:20010888180f157f0000000000000001
6hostb.local.net:20010888180f157f0000000000000002
6firewall.local.net:20010888180f157f0000000000000254
6hostasitelocal.local.net:fec0000000000000020024fffec06936
6hostbsitelocal.local.net:fec0000000000000250bffffe124c5b
6firewallsitelocal.local.net:fec0000000000000024095fffe161d42
```

Note

Notice that the IPv6 address needs to be fully specified, no abbreviation with colons ("::") is allowed.

Limitations & known problems

The known limitations and problems with these applications are listed below. If you happen to have a solution for these issues, please let us know.

- IPv6 transport support is experimental. The **dnscache** and **tinydns** daemons are bound to IPv4-mapped IPv6 addresses, e.g. `::ffff:192.168.1.254`

inetd

Overview

Description	USAGI inetd daemon
Source	Prepatched source tarball by the USAGI project [http://www.linux-ipv6.org]
Version	0.17 + USAGI IPv6 patches
Package	`root.lrp`

Configuration

Use the keywords **tcp6** and **udp6** in `/etc/inetd.conf` to let inetd listen on IPv6 sockets. For example, if you want to weblet to be accessible via IPv6 you should configure `/etc/inetd.conf` as follows:

```
(...)
www     stream  tcp6    nowait  sh-httpd    /usr/sbin/tcpd  /usr/sbin/sh-httpd
(...)
```

Limitations & known problems

The known limitations and problems with this application are listed below. If you happen to have a solution for these issues, please let us know.

- It is not possible to let inetd listen to the same portnumber for IPv4 and IPv6 sockets. Tests with the **tcp46** keyword have failed. Also putting two configuration lines in `/etc/inetd.conf`, one with the **tcp(4)** and one with the **tcp6** keyword, has not given the desired result.

pppd

Overview

Description	PPP daemon
Source	www.samba.org/ppp/ [http://www.samba.org/ppp/]
Version	2.4.1
Package	`ppp.lrp`

Configuration

To be provided.

Limitations & known problems

The known limitations and problems with this application are listed below. If you happen to have a solution for these issues, please let us know.

- To be provided.

snmpd

Overview

Description	SNMP daemon from Net-SNMP
Source	www.net-snmp.org [http://www.net-snmp.org]
Version	5.0.8
Package	`netsnmpd.lrp`

Configuration

This section only describes how to use the IPv6 features of netsnmpd. For general configuration issues refer to the documentation on the net-snmp [http://www.net-snmp.org] site.

Default snmpd listens only to IPv4 sockets. Extra parameters can be used on startup of the daemon to make it listen to both IPv4 and IPv6 sockets. One of the current shortcommings is that the IPv4 and IPv6 port numbers on which the daemon listens may not be the same. For example to let snmpd listen on port 161 for IPv4 and on port 6161 for IPv6 edit `/etc/init.d/snmpd` as follows.

```
(...)
# Set cli options here
OPTIONS="udp:161,udp6:6161"
OPTIONS=${OPTIONS:+-- $OPTIONS}
(...)
```

Restart the daemon with the command **/etc/init.d/snmpd restart**. You can check if the snmpd daemon is really listening to both sockets with the following command:

```
# netstat -na
Active Internet connections (servers and established)
```

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State
(...)
udp        0      0 0.0.0.0:161             0.0.0.0:*
udp        0      0 :::6161                 :::*
(...)
```

Limitations & known problems

The known limitations and problems with this application are listed below. If you happen to have a solution for these issues, please let us know.

- You can get the daemon to listen to IPv4 sockets as well as IPv6 sockets, but not on the same UDP ports.
- Not the full IPv6 MIB can be retrieved via commands like **snmpwalk**, only the following object instances are returned:

```
.iso.3.6.1.2.1.55.1.1.0 = 1
.iso.3.6.1.2.1.55.1.2.0 = 64
.iso.3.6.1.2.1.55.1.3.0 = Gauge32: 3
.iso.3.6.1.2.1.55.1.5.1.2.1 = "lo"
.iso.3.6.1.2.1.55.1.5.1.2.3 = "eth0"
.iso.3.6.1.2.1.55.1.5.1.2.4 = "eth1"
.iso.3.6.1.2.1.55.1.5.1.3.1 = OID: .ccitt.0
.iso.3.6.1.2.1.55.1.5.1.3.3 = OID: .ccitt.0
.iso.3.6.1.2.1.55.1.5.1.3.4 = OID: .ccitt.0
.iso.3.6.1.2.1.55.1.5.1.4.1 = Gauge32: 16436
.iso.3.6.1.2.1.55.1.5.1.4.3 = Gauge32: 1500
.iso.3.6.1.2.1.55.1.5.1.4.4 = Gauge32: 1500
.iso.3.6.1.2.1.55.1.5.1.8.1 = ""
.iso.3.6.1.2.1.55.1.5.1.8.3 =  Hex: 00 40 95 1A 14 F4
.iso.3.6.1.2.1.55.1.5.1.8.4 =  Hex: 00 40 95 1A 14 70
.iso.3.6.1.2.1.55.1.5.1.9.1 = 1
.iso.3.6.1.2.1.55.1.5.1.9.3 = 1
.iso.3.6.1.2.1.55.1.5.1.9.4 = 1
.iso.3.6.1.2.1.55.1.5.1.10.1 = 1
.iso.3.6.1.2.1.55.1.5.1.10.3 = 1
.iso.3.6.1.2.1.55.1.5.1.10.4 = 1
```

sshd

Overview

Description	Secure shell daemon
Source	www.openssh.org [http://www.samba.org/ppp/]
Version	3.7.1p1
Packages	`sshd.lrp`, `libm.lrp`, `libcrpto.lrp`

Configuration

This section only describes how to use the IPv6 features of sshd. For general configuration issues refer to the documentation on the OpenSSH [http://www.openssh.org] site.

Sshd is compiled with TCP-wrappers support. Thus means that the `hosts.allow` and `hosts.deny` files are used for acces control purposes. If you want to access the ssh daemon with an IPv6 enabled client such as PuTTY [http://unfix.org/projects/ipv6/], then you need to specifiy the IPv6 address for the single client or the prefix for more clients in the same subnet.

The following is an example how to modify `/etc/hosts.allow` to allow all clients with a site-local address:

```
(...)
ALL: [fec0::/64]
(...)
```

Limitations & known problems

The known limitations and problems with this application are listed below. If you happen to have a solution for these issues, please let us know.

- To be provided.

Chapter 11. freenet6.lrp - access for tunnel broker freenet6

K.-P. Kirchdörfer <kapeka at users.sourceforge.net>

Revision History
Revision 0.1 2004-03-28 kp
Initial Document

Introduction

The `freenet6.lrp` package provides tspc - tunnel setup protocol client - used by the ipv6 tunnel broker freenet6 [http://www.freenet6.net] to give you access to 6bone. It will allow you to connect to 6bone with an ipv6 address for your LEAF router or a complete ipv6 subnet for your network. It will also setup a tunnel from your LEAF router to the ipv6 network. This works with fixed as well as with dynamic ipv4 addresses (Dial-up links, ADSL links etc...)

tspc has been outlined in the Internet draft draft-vg-ngtrans-tsp-01 [http://www.freenet6.net/draft-tsp.shtml], another explanation and test of this approach can be found here [http://www.iihe.ac.be/internal-report/2003/stc-03-02.pdf].

A very good introduction, setup instruction for Debian packages and recommended reading is available from Jean Marc Liotier - Jim's insignificant LAN IPv6 global connectivity HOWTO [http://www.ruwenzori.net/ipv6/Jims_LAN_IPv6_global_connectivity_howto.html]. Because we just adapted the Debian package for LEAF Bering-uClibc, almost everything fit's to the `freenet6.lrp` as well.

Declare the freenet6.lrp package

Download the freenet6.lrp [http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/leaf/bin/packages/uclibc-0.9/20/freenet6.lrp?rev=HEAD&content-type=application/octet-stream] package and copy the package to your Bering-uClibc diskette.

Boot a Bering-uClibc floppy image. Once the LEAF menu appears get access to the linux shell by (q)uitting the menu. Edit the `lrpkg.cfg` (pre Bering-uClibc-2.2.0) or `leaf.cfg` (Bering-uClibc-2.2.0 onwards) file and add `freenet6.lrp` in the list of packages to be loaded at boot. Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-install.html] to learn how to do that.

Obtain an (authenticated) tunnel or a whole subnet

If you just need an ipv6 address for LEAF router, you don't have to do anything and can skip step 4. But in most cases you like to have an authenticated tunnel (esp. with dynamic ipv4 adddress) or obtain a /48 prefix delegation for your LAN and probably to subnet a few more ipv6 networks.

To get an authenticated tunnel or a /48 prefix delegation, go to www.freenet6.net/register.shtml [http://www.freenet6.net/register.html] and create an account. Accounts are mandatory on Freenet6 if you want an authenticated tunnel or a /48 IPv6 prefix delegation. The authenticated tunnel provides one single and permanent IPv6 address to a node in spite of Ipv4 address changes. The /48 IPv6 prefix delegation is how you get a bunch of addresses for those hosts inside your LAN.

Please note that the "username" on the registration page is misleading - it will be the name of your machine/router.

After registration you will receive an email with your "username" and password form freenet6.net.

Configure freenet6

Edit `/etc/freenet6/tspc.conf`: Make sure that the values assigned to userid and passwd are the ones that you got by mail from Freenet6. Also add the following options if you need a /48 prefix delegation:

```
host_type=router
prefixlen=48
```

and in our example we changed

```
if_tunnel=tunFN
```

Change is due to a remark from Dr. Peter Bieringer: (see: http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/configuring-ipv6to4-tunnels.html

[This is now deprecated because using the generic tunnel device sit0 doesn't let specify filtering per device.]

see: http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/configuring-ipv6to4-tunnels.html

Note

Backup freenet6.lrp.

Configure the firewall

Configure shorewall

You need to allow the 6to4 tunnel traffic from the Bering-uClibc Firewall to the tunnelbroker freenet6.org. This traffic uses IP protocol 41.

With Shorewall 1.4.3 and later you can add a 6to4 tunnel definition in the `/etc/shorewall/tunnels` file. In the examples below it is assumed that your firewall zone is called "fw" and the Internet zone "net". The gateway address in 6to4tunnel is your tunnel end-point at freenet6.org.

```
#TYPE    ZONE   GATEWAY         GATEWAY ZONE
(...)
6to4     net    206.123.31.115
(...)
```

More information on Shorewall and 6to4 tunnels can be found in the Shorewall documentation.

Additionally you have to open port 3653 for tspc used with freent6.org in `/etc/shorewall/rules`:

```
# tspc  - tunnel setup protocol
ACCEPT        fw        net:206.123.31.115          tcp      3653
ACCEPT        net:206.123.31.115      fw           tcp      3653
```

Note

Backup shorwall.lrp.

Configure 6wall

Add the tunnel interface to 6wall `/etc/6wall/interfaces6`:

```
#ZONE   INTERFACE       OPTIONS
#
net     tunFN           nositelocal
loc     eth1
#
```

For more information about 6wall please look at:

http://leaf.sourceforge.net/devel/dorus/sixwall.html

Note

Backup 6wall.lrp.

Using radvd

Manual or automatic radvd configuration

freenet6 is supposed to take care of configuring `radvd` by writing `radvd.conf` for you and restarting `radvd` automatically. Andreas Rottmann, the Debian maintainer of radvd, decided that it should not let freenet6 rewrite the entire radvd configuration file lest manual modifications by the administrator be overwritten. He asked if it would be possible to have a way for freenet6 to change the advertised prefix of radvd without rewriting the whole config file. But Nathan Lutchansky (radvd programmer) answered in substance that administrators, who do not want to risk their modifications overwritten should handle radvd.conf manually. In typical Debian fashion Andreas Rottman probably believes he is better safe than sorry, and so he commented out the parts of /etc/freenet6/setup.sh that deal with rewriting radvd.conf - and we decided to follow his approach - just to be on the safe side as well.

Automatic radvd configuration

If you think you have an easy-to-use setup you can uncomment the section in `/etc/freenet6/setup.sh` shown in the screenshot below - at least we saw no problems with that automatic rewriting and restart of radvd in a freenet6 only setup.

```
#   Display 1 "Create new $rtadvdconfigfile"
#   echo "##### rtadvd.conf made by TSP ####" > "$rtadvdconfigfile"
#   echo "interface $TSP_HOME_INTERFACE" >> "$rtadvdconfigfile"
#   echo "{" >> "$rtadvdconfigfile"
#   echo "  AdvSendAdvert on;" >> "$rtadvdconfigfile"
```

```
#   echo " prefix $TSP_PREFIX:0001::/64" >> "$rtadvdconfigfile"
#   echo " {" >> "$rtadvdconfigfile"
#   echo " AdvOnLink on;" >> "$rtadvdconfigfile"
#   echo " AdvAutonomous on;" >> "$rtadvdconfigfile"
#   echo " };" >> "$rtadvdconfigfile"
#   echo "};" >> "$rtadvdconfigfile"
#   echo "" >> "$rtadvdconfigfile"
#   /etc/init.d/radvd stop
#   if [ -f $rtadvdconfigfile ]; then
#      KillProcess $rtadvdconfigfile
#      Exec $rtadvd -C $rtadvdconfigfile
#      Display 1 "Starting radvd: $rtadvd -C $rtadvdconfigfile"
#   else
#      echo "Error : file $rtadvdconfigfile not found"
#      exit 1
#   fi
```

Note

You need to install `mawk.lrp` to use automatic `radvd` configuration.

Manual radvd configuration

For a manual configuration of `radvd` you need to know the subnet assigned to you by freenet6. After you have configured freenet6 and rebooted you'll find the assigned subnet with:

 # ip -6 addr show | grep 3ffe

The result will look like:

```
inet6 3ffe:bc0:b40:1::1/64 scope global
inet6 3ffe:bc0:8000::3497/128 scope global
```

The first line must be added to `radvd.conf` as prefix - like here:

```
interface eth1
{
   AdvSendAdvert on;
   prefix 3ffe:0bc0:0b40:0001::/64

   {
        AdvOnLink on;
        AdvAutonomous on;
   };
};
```

Save `radvd.lrp` and restart `/etc/init.d/radvd`.

Chapter 12. Zebra configuration

Eric Spakman <e.spakman at inter.nl.net>

Revision History
Revision 0.1 2003-08-17 espakman
Initial version

Overview

Zebra is a routing daemon. That means, it will send routing requests and receive similar requests from neighbor routers, and eventually update your routing tables accordingly. Zebra provides TCP/IP based routing services with routing protocols support such as RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4, and BGP-4+.

RIP and OSPF are "internal" routing protocols, whereas BGP is an "external" routing protocol. Internal protocols are designed for use in LANs, within a global administrative scope. External protocols are designed for use in WAN, and BGP is specifically designed for Internet use. Zebra also supports special BGP Route Reflector and Route Server behavior. In addition to traditional IPv4 routing protocols, Zebra also supports IPv6 routing protocols.

There are five routing daemons in use, and there is one manager daemon. These daemons may be located on separate machines from the manager daemon.

The routing daemons are:

- ripd, ripngd, ospfd, ospf6d, bgpd

The manager daemon is:

- zebra

Configuring Zebra

Zebra's architecture includes an O/S dependant application, whose role is to deal with network interface configuration, routing table updates, and other kernel stuff ; and O/S independant routing processes, communicating thru sockets with the Zebra core. You will have to choose which daemons you want to run by loading and configuring the appropiate packages.

If you want to activate the processes zebra and bgpd, edit bgpd.conf (note that lines beginning with ! are comments). No configuration is necessary in zebra.conf, but you should edit bgpd.conf to include the following lines :

router bgp ASN bgp router-id ROUTERID network 192.168.A.B/M network 192.168.C.D/N neighbor 192.168.P.Q remote-as REMOTEASN

Where ASN is your Autonomous System Number (it will look like a number above 65000, and will be given when you ask for it to the tunnel's maintener), ROUTERID is a dummy IP address (it can be 5.4.3.2 if you like, it's just an identifier). You should specify your network entries with respect to your allocated IP addresses ranges. For instance, if you told the maintener that you would use 192.168.93.1 thru 192.168.93.127, specify 192.168.93.0/25 . You can specify multiple network routes. If you are part

of the backbone (you will be told if that is the case), you should export a host route (192.168.0.X/32). The neighbor IP address and ASN will be given to you by your maintener.

After editing configuration files, do a backup and start zebra and bgpd ("svi zebra start" and "svi bgpd start" or reboot the router). The routes should appear within your kernel routing table ("ip route").

You need to open the appropiate ports in shorewall (fw <-> loc for internal and fw <-> net for external routing protocols) to make the routing exchange possible.

Configuring Zebra with telnet

The individual daemons also provide a vty interface for Cisco like configuration. There are two ways of doing this: by telnetting to localhost (this method is not further described, because Bering-uClibc doesn't provide a telnet client due to securrity reasons) and by telnetting to the router with telnet from a client machine. Opening ports on the firewall is always a securitty risk, so only do this is you trust your local-net.

Open the zebra port and one of more of the daemon ports in shorewall (loc to net):

Table 12.1. Daemon ports:

zebra	2601/tcp	# zebra vty
ripd	2602/tcp	# RIPd vty
ripngd	2603/tcp	# RIPngd vty
ospfd	2604/tcp	# OSPFd vty
bgpd	2605/tcp	# BGPd vty
ospf6d	2606/tcp	# OSPF6d vty

First configure the routing daemons as described in the previous section, additional you can set a password and a Hostname as described below:

Password: The password is like a standard user password, and think about the "enable password" like the "root password" of an UNIX box. If you don't put an "enable password", it won't be necessary (id est, empty password).

Hostname: You can also configure hostnames : If you use foo-zebra as hostname the router's name is "foo" for zebra process, and foo-bgpd for the bgpd process. The hostname only influences the command prompt when you connect to a router (with telnet firewall bgpd for instance).

After backup and starting of the different daemons, you can connect to them with: telnet <firewall> <port or name>

Check the BGP configuration: Connect to your BGP routing process : telnet firewall bgpd ; you will be prompted for a password. At the prompt, issue enable and give your enable password. Then, show ip bgp will show BGP routes ; show ip bgp summary will show neighbors state. The latter should look like this: `firewall-bgpd# sh ip bg su Neighbor V AS MsgRcvd MsgSent TblVer InQ OutQ Up/Down State/Pref 192.168.168.168 4 65168 14062 13971 0 0 0 1d23h59m 6 192.168.192.1 4 65301 6110 6250 0 0 0 2d00h01m 1 firewall-bg-pd#`

Notice that you can abbreviate commands. If you see big variations between MsgRcvd and MsgSent, that may be a hint of network failures.

You have to backup the configuration with both "write" in the vty and a backup of the package on the Bering-uClibc firewall itself.

Links

http://www.zebra.org/

http://skaya.enix.org/vpn/zebra.html

Chapter 13. Using SNMP and RRD to monitor your LEAF system

Eric de Thouars <dorus at users.sourceforge.net>

Revision History
Revision 0.1 2004-10-18 ET
Initial Document

Introduction

Objectives

In this chapter it is described how you can monitor the performance of your LEAF system in near real-time using SNMP and RRD.

Overview of the setup described here

The setup that is described here assumes that you have at least two systems, the LEAF system that you want to monitor and a system that will collect, store and present the performance data. In the rest of this chapter these systems will be indicated as the **LEAF** system and the **RRD** system.

The RRD system will query the LEAF system on regular intervals via snmp. The collected data is stored in an RRD database. The performance data can be presented in a number of ways. Here it will be presented using a webserver with php scripts containing rrdtool functions.

The setup and configuration of the LEAF system is simple compared to the setup and configuration of the RRD system. All that is needed on the LEAF system is an SNMP agent. The RRD system can be made as simple or advanced as desired by the user. At least the following functionalities must be present on the RRD system

- SNMP client to query the SNMP agent in the LEAF system
- Database to store and retrieve the measured data

The SNMP client and agent functions in this sample are provided by the Net-SNMP package. The database for storing the measured data is based on RRDTool. In the next sections a short overview of these toolkits is given.

About Net-SNMP

The Net-SNMP [http://net-snmp.sourceforge.net/] toolkit provides a suite of client and server applications that communicate with each other using the Simple Network Management Protocol (SNMP).

One of the server applications is `snmpd`, which is an SNMP Agent. `snmpd` listens for SNMP requests. A typical SNMP agent allows a client to query information about the device running the SNMP agent. Some devices also allow configuration to be set via SNMP.

The Net-SNMP agent can be built to monitor things such as network traffic,disk space, disk IO, CPU usage and more.

Next to the server part, the client part is needed. In this example the Perl libraries of Net-SNMP are used for the client part. Perl scripts on the RRD system are used to collect the performance data from the LEAF system.

About RRDTool

RRD is the Acronym for Round Robin Database. RRD is a system to store and display time-series data (i.e. network bandwidth, machine-room temperature, server load average). It stores the data in a very compact way that will not expand over time, and it presents useful graphs by processing the data to enforce a certain data density. It can be used either via simple wrapper scripts (from shell or Perl) or via front-ends that poll network devices and put a friendly user interface on it.

In the rest of this document it is assumed that you have at least read the "RRD Beginners Guide" and the "RRD Tutorial" from the RRDTool documentation [http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/tutorial/]page.

Configure the LEAF system

Load netsnmpd package

Add the `netsnmpd`, `libsnmp` and `libm` packages to the packages list. If you don't know how to do this, check the section "Adding and removing packages [buci-lrpkg.html]" from the Bering-uClibc Installation Guide.

Either reboot the system or load the new packages manually.

Configure the snmp daemon

Edit the configuration file `/etc/snmp/snmpd.conf`. A sample configuration is given below. This sample does not contain all the helpful comments from the original configuration file, so I suggest you use this to edit your existing configuration file.

```
#
# snmpd.conf
#

syscontact  "Root <zaphod@heart.of.gold>"
syslocation  "At the end of the Universe"
sysname leafhost
sysservices 15

rocommunity  <your_community_string>     default
com2sec      readonly   default    <your_community_string>
group        RO_Group   usm   readonly
group        RO_Group   v1    readonly
group        RO_Group   v2c   readonly
view         all    included   .1

access  RO_Group   ""       any        noauth     exact  all     none   none

#
```

Now backup the `netsnmpd` package and (re)start start snmpd with **svi snmpd restart**.

Configure the RRD machine

Prerequisites

For the examples given here the following items must be installed on the RRD system.

- Perl::SNMP - Net-SNMP module for Perl (source: netsnmp.sourceforge.net [http://netsnmp.sourceforge.net])
- Perl::RRDs - RRDTool module for Perl, use perl-shared not perl-piped (source: people.ee.ethz.ch/~oetiker/webtools/rrdtool [http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/])
- Apache with PHP4 - Webserver for presentation of the performance data (source: www.apache.org [http://www.apache.org], www.php.org [http://www.php.org])
- Php4-rrdtool - RRDTool module for PHP4 (source: www.joeym.net [http://www.joeym.net/])

For the rest of this document it is assumed that you are running Linux on your RRD system. This is not the only possible option, the necessary items are also available for other types of systems. It is beyond the scope of this document to describe where to get the above mentioned items precompiled for your system and how to install them. Refer to the documentation of your distribution and/or the documentation of the individual sources for more information.

Collecting and storing performance data

Introduction

In this chapter the terms **collector** and **database** will be used frequently. The collector is the script that queries the LEAF system via SNMP and stores the retrieved values in a database, in this case an RRD database.

An RRD database can be defined to contain all sorts of information, datasets, in any combination you like. It is in general good practice to keep information of different types in different databases, but you'll have to find out for yourself which dataset definition will give you the most flexible solution for your situation.

In the following examples two datasets will be defined, one for network traffic statistics and one for cpu-load.

Personally I like to structure the RRD related directories in such a way that there is a clear distinction between collectors and databases, and also between databases belonging to different hosts. In these examples the following directory structure is assumed:

```
/home/rrd/
       |
       +--- collectors/
       |
       +--- databases/
                 |
                 +--- leafhost/
                 |
                 +--- host2/
                 |
                   ... etc ...
```

After defining a database and creating the corresponding collector, the collector must be scheduled to run at regular intervals. This must be done for each collector/database. Cron is your friend here. An option that I favor myself is to have only one entry in `/etc/crontab`. This entry calls the overall collector script, which in turn calls each of the individual collector scripts. This avoids that for each new collector the system crontab file must be edited. In this case your `/etc/crontab` would have the following entry:

```
# /etc/crontab

...

# overall collector script
*/5 *   * * *   rrd    /home/rrd/collectors/collect-all

#
```

Note

Maybe trivial, but the above applies to the crontab file on the RRD system and NOT to the crontab file of the LEAF system.

This means that the overall collector script is started every 5 minutes. The overall collector file `/home/rrd/collectors/collect-all` could look like:

```
#!/bin/sh
# Overall collector script

# Script for collecting interface statistics
/home/rrd/collectors/interface.pl

# Script for collecting cpu load
/home/rrd/collectors/cpuload.pl
```

Example 1: network traffic

Define the RRD database

If the number of interfaces on the LEAF system is fixed and will never change, you may choose to keep the traffic statistics of both interfaces in one database. If not, it's probably easier to define a database per interface. This makes it easier extend your RRD system for more interfaces that you may get on your LEAF system. Here a database for only one interface is created.

To create a new database, go to the data directory for the targeted host and create the dataset with the options as described below:

```
cd /home/rrd/databases/leafhost
rrdtool create eth0.rrd \
        -step 300 \
        DS:bytes_in:COUNTER:600:U:U \
        DS:bytes_out:COUNTER:600:U:U \
        RRA:AVERAGE:0.5:1:864 \
        RRA:AVERAGE:0.5:6:672 \
        RRA:AVERAGE:0.5:24:744 \
        RRA:AVERAGE:0.5:288:730
```

This has created a new database named `eth0.rrd` which expects new data every 300 seconds (step size). This is exactly the same as the schedule defined in the crontab file above.

The database contains two datasets, i.e. `bytes_in` and `bytes_out`, both of the type COUNTER.

Three round robin archives are defined containing avaraged values:

- 864 samples of 1 step (5 seconds). This is a period of 3 days. Since the step size is one the actual value is stored and no average is calculated.
- 672 averaged samples over 6 steps (30 minutes). This is a period of 2 weeks.
- 744 averaged samples over 24 steps (2 hours). This a period of 2 weeks.
- 730 averaged samples over 288 steps (1 day). This is a period of 2 years.

Create the collector

The data that can be retrieved from an SNMP agent is defined in a Management Information Base MIB). The objects in the MIB containing the interface traffic counters that are necessary for this example are:

- .iso.org.dod.internet.mgmt.mib-2.interfaces.ifNumber = .1.3.6.1.2.1.2.1
- .iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr = .1.3.6.1.2.1.2.2.1.2
- .iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifInOctets = .1.3.6.1.2.1.2.2.1.10
- .iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOutOctets = .1.3.6.1.2.1.2.2.1.16

In the sample script below the LEAF system is queried for the number of interfaces. The correct interface is selected based on the interface name and then the counters for bytes_in and bytes_out are read. Finally this information is stored into the database.

```
#!/usr/bin/perl

# interface.pl

use SNMP;
use RRDs;

$oid_ifNumber    = ".1.3.6.1.2.1.2.1";
$oid_ifDescr     = ".1.3.6.1.2.1.2.2.1.2";
$oid_ifInOctets  = ".1.3.6.1.2.1.2.2.1.10";
$oid_ifOutOctets = ".1.3.6.1.2.1.2.2.1.16";

$database = "/home/rrd/databases/leafhost/eth0.rrd";


#
# Open snmp session and get interface data
#
$session = new SNMP::Session(
                        DestHost  => "leafhost",
                        Community => "<your_community_string>",
                        Version   => '2');
die "SNMP session creation error: $SNMP::Session::ErrorStr" unless (defined $sessi
```

```
$numInts = $session->get($oid_ifNumber . ".0");

for $i (1..$numInts) {
    $name = $session->get($oid_ifDescr . "." . $i);
    if ( $name eq "eth0" ) {
        $in = $session->get($oid_ifInOctets . "." . $i);
        $out = $session->get($oid_ifOutOctets . "." . $i);
    }
}

die $session->{ErrorStr} if ($session->{ErrorStr});


#
# Update the database
#
RRDs::update ($database, "N:".$in.":".$out);
my $Err = RRDs::error;
die "Error while updating: $Err\n" if $Err;

#
```

Ofcourse this is only an example. You can use this to extend it to your own needs.

Example 2: cpu load

Define the RRD database

On Linux systems three types of cpu load (process time) exist, i.e. user, system, nice and idle. We will now define a database in which to store this information.

```
cd /home/rrd/databases/leafhost
rrdtool create cpuload.rrd \
        --step 300 \
        DS:user:COUNTER:600:0:100 \
        DS:system:COUNTER:600:0:100 \
        DS:nice:COUNTER:600:0:100 \
        DS:idle:COUNTER:600:0:100 \
        RRA:AVERAGE:0.5:1:864 \
        RRA:AVERAGE:0.5:6:672 \
        RRA:AVERAGE:0.5:24:744 \
        RRA:AVERAGE:0.5:288:730
```

The definition of this database has much in common with the previous database. Now four datasets have been defined instead of two. The definition of the round robin archives is the same.

Create the collector

The cpu load information is represented by the following objects in the MIB:

- .iso.org.dod.internet.private.enterprises.ucdavis.systemStats.ssCpuRawUser = .1.3.6.1.4.1.2021.11.50
- .iso.org.dod.internet.private.enterprises.ucdavis.systemStats.ssCpuRawNice = .1.3.6.1.4.1.2021.11.51

- .iso.org.dod.internet.private.enterprises.ucdavis.systemStats.ssCpuRawSystem = .1.3.6.1.4.1.2021.11.52
- .iso.org.dod.internet.private.enterprises.ucdavis.systemStats.ssCpuRawIdle = .1.3.6.1.4.1.2021.11.53

And this information can be retrieved and stored with the following script:

```
#!/usr/bin/perl

# cpuload.pl

use SNMP;
use RRDs;

$oid_ssCpuRawUser    = ".1.3.6.1.4.1.2021.11.50";
$oid_ssCpuRawSystem  = ".1.3.6.1.4.1.2021.11.51";
$oid_ssCpuRawNice    = ".1.3.6.1.4.1.2021.11.52";
$oid_ssCpuRawIdle    = ".1.3.6.1.4.1.2021.11.53";

$database = "/home/rrd/databases/leafhost/cpuload.rrd";


#
# Open snmp session and get interface data
#
$session = new SNMP::Session(
                        DestHost  => "leafhost",
                        Community => "<your_community_string>",
                        Version   => '2');
die "SNMP session creation error: $SNMP::Session::ErrorStr" unless (defined $sessi

$cpuUser   = $session->get($oid_ssCpuRawUser . ".0");
$cpuSystem = $session->get($oid_ssCpuRawSystem . ".0");
$cpuNice   = $session->get($oid_ssCpuRawNice . ".0");
$cpuIdle   = $session->get($oid_ssCpuRawIdle . ".0");


#
# Update the database
#
RRDs::update ($database, "N:".$cpuUser.":".$cpuSystem.":".$cpuNice.":".$cpuIdle);
my $Err = RRDs::error;
die "Error while updating: $Err\n" if $Err;

#
```

Retrieving and presenting performance data

Introduction

After you finished the scripts and the overall collector has been called a few times by cron, it's time to make some graphics.

The follwoing assumptions are made with respect to the configuration of the webserver:

- An alias `/images/` is defined for `/var/www/images/`
- The images directory has a subdirectory `rrdimg` in which the rrd graphs will be created

For ease of reuse a separate php file is used in which the generic functions for drawing graphs are defined. This file is included by the other scripts.

Example 1: network traffic

First a file `graphs.php` is defined that contains the functions to draw the graphs.

```
<?php

## graphs.php
##
## A set of php functions to create rrd graphs


function interface ($start)
{
    $dataset = "/home/rrd/databases/leafhost/eth0.rrd";
    $imgfile = "eth0$start.gif";

    $opts = array( "--start", "$start",
        "--vertical-label", "Bytes/sec",
        "--width", "400",
        "DEF:in=$database:bytes_in:AVERAGE",
        "DEF:out=$database:bytes_out:AVERAGE",
        "LINE2:in#00ff00:In",
        "LINE2:out#ff0000:Out"
    );

    make_graph ($imgfile, $opts);
}


function make_graph ($file, $options)
{
    $ret = rrd_graph("/var/www/images/rrdimg/$file", $options, count($options));

    ## if $ret is an array, then rrd_graph was successful
    ##
    if ( is_array($ret) ) {
        echo "<img src=\"/images/rrdimg/$file\" border=0>";
    }
    else {
        $err = rrd_error();
        echo "<p><b>$err</b></p>";
    }
}

?>
```

Then the actual page that contains the network traffic graphs can be created.

```
<html>
    <head>
        <title>Interface statistics</title>
    </head>
    <body>
        <h1>Interface statistics</h1>
<?php
        require "graphs.php";

        print "<h2>Daily graph</h2>\n";
        interface ("-1d");
```

```
        print "<h2>Weekly graph</h2>\n";
        interface ("-1w");
        print "<h2>Monthly graph</h2>\n";
        interface ("-1m");
?>
    </body>
</html>
```

Now fire-up your browser and access the page that you just created. Sit back and enjoy !!

Example 2: cpu load

First we add a function to draw cpuload garphs to the file `graphs.php`.

```
<?php

## functions.php
##
## A set of php functions to create rrd graphs

...

function cpuload ($start)
{
    $database = "/home/rrd/databases/leafhost/cpuload.rrd";
    $imgfile = "cpu$start.gif";

    $opts = array( "--start", "$start",
        "--vertical-label", "Load (%)",
        "--width", "400",
        "DEF:user=$database:user:AVERAGE",
        "DEF:nice=$database:nice:AVERAGE",
        "DEF:system=$database:system:AVERAGE",
        "AREA:system#00ffff:System",
        "STACK:user#00ff00:User",
        "STACK:nice#0000ff:Nice",
    );

    make_graph ($imgfile, $opts);
}

?>
```

And then the actual CPU load page is created. This is almost too easy `;-)`

```
<html>
    <head>
        <title>CPU Load statistics</title>
    </head>
    <body>
        <h1>CPU Load statistics</h1>
<?php
        require "graphs.php";

        print "<h2>Daily graph</h2>\n";
        cpuload ("-1d");
        print "<h2>Weekly graph</h2>\n";
        cpuload ("-1w");
        print "<h2>Monthly graph</h2>\n";
```

```
        cpuload ("-1m");
?>
    </body>
</html>
```

Chapter 14. Increasing ip_conntrack_max and hashsize

Eric Spakman <espakman at users.sourceforge.net>
K.-P. Kirchdörfer <kapeka at users.sourceforge.net>

Revision History
Revision 0.2 2004-10-17 kp/es
sysctl.conf
Revision 0.1 2004-05-01 kp
Initial Document

Introduction

Sometimes the defaults for netfilter conntrack (and thus NAT) does not fit the needs of a high-loaded firewall.

The default sizes for ip_conntrack_max and hashsize (the number of seperate connections that can be tracked, and the size of the hash table that keeps track of them, respectively) defaults to a percentage of your total memory size. This percentage is geared towards a 'general use' workstation with lots more memory (and fewer connections to track) than a typical special-purpose firewall box. The hash table works much better when it's size is a prime number.

Beginning with Bering-uClibc 2.2 it is possible to tweak performance, while loading the ip_conntrack module (in `/etc/modules`).

Configuration

You can set the ip_conntrack_max parameter by using `sysctl.conf` (listed under System configuration), some examples are provided in this file.

```
#
# /etc/sysctl.conf - Configuration file for setting system variables
#
# Examples:
#
# Set the ip_conntrack limit
#net.ipv4.netfilter.ip_conntrack_max=65000
#
# Set the arp limit
#net.ipv4.neigh.default.gc_thresh1=16
#net.ipv4.neigh.default.gc_thresh2=256
#net.ipv4.neigh.default.gc_thresh3=2048
```

Sysctl is used to modify kernel parameters at runtime. The parameters available are those listed under `/proc/sys/`. The variable is the key to read from. An example is kernel.ostype. The '/' seperator is also accepted in place of a '.'. To set a key, use the form variable=value, where 'variable' is the key and 'value' is the value to set it to.

The hashsize parameter can be set while loading the ip_conntrack module (this is done in the modules package):

ip_conntrack hashsize=$HASHSIZE where $HASHSIZE is an integer.

Links

Detailed instructions can be found in the following document: http://www.wallfire.org/misc/netfilter_conntrack_perf.txt

A handy table of prime numbers good for hash table sizes can be found at PlanetMath: http://planetmath.org/encyclopedia/GoodHashTablePrimes.html

Thanks

The idea and the information in this chapter is originally from a mail of Charles Steinkuehler sent to leaf-user@lists.sourceforge.net.

Chapter 15. Using keepalived with LEAF Bering-uClibc

K.-P. Kirchdörfer <kapeka at user.sourceforge.net>
Peter Mueller <peter at sidestep.com>

Revision History
Revision 0.1 2004-10-14 kp
Initial version

Objectives

Keepalived is a high-availability and load-balancing tool. Using keepalived, virtual IPs and Linux Virtual Server and Virtual Router Redundancy setups can be managed very effectively between two or more hosts. From the Keepalived site: “The main goal of the keepalived project is to add a strong & robust keepalive facility to the Linux Virtual Server project. his project is written in C with multilayer TCP/IP stack checks. Keepalived implements a framework based on three family checks: Layer3, Layer4 & Layer5/7. This framework gives the daemon the ability of checking a LVS server pool states. When one of the server of the LVS server pool is down, keepalived informs the linux kernel via a setsockopt call to remove this server entry from the LVS topology. In addition keepalived implements an independent VRRPv2 stack to handle director failover. So in short keepalived is a userspace daemon for LVS cluster nodes healthchecks and LVS directors failover.”

In our case we are mostly interested in the Virtual Router Redundancy Protocol (VRRP) part. A comprehensive introduction into VRRP can be found in the IBM Redpaper "Virtual Router Redundancy Protocol (VRRP) on VM Guest LANS" (see Link section below).

Load the keepalived and additionally required packages

To install keepalived add kpalived.lrp and the additionally required packages libpopt.lrp, libssl.lrp, libcrpto.lrp to leaf.cfg. Check the Bering-uClibc Installation Guide [http://leaf.sourceforge.net/doc/guide/buci-lrpkg.html] to learn how to do that.

Configuration

```
! Configuration File for keepalived

global_defs {
   notification_email {
     sysadmin@yourcompany.com
   }
   notification_email_from keepalived@yourcompany.com
   smtp_server 192.168.1.200
   smtp_connect_timeout 30
   lvs_id LVS1
}

! sync groups bond instances together.  they are tricky,
! so read the documentation and/or mailing lists before using them.

!vrrp_sync_group LVS1 BACKUP {
```

```
!    VI_1
!    VI_2
!    }

vrrp_instance VI_1 {
    state MASTER
    track_interface {
        eth0
    }
    interface eth2 # interface to send multicast heartbeat on
    virtual_router_id 51
    priority 150  # the highest priority is the master
    advert_int 2  # rate of multicast heartbeats (seconds)
    authentication {
        auth_type PASS # don't use IPSEC, it is buggy
        auth_pass SECRETPASS
    }
    virtual_ipaddress {
        192.168.1.210  # list as many IPs as you want, one perline.  see SYNOPSIS
    }
}

vrrp_instance VI_2 {
    state SLAVE
    track_interface {
        eth0
    }
    interface eth2
    virtual_router_id 52
    priority 100
    advert_int 2
    authentication {
        auth_type PASS
        auth_pass SECRETPASS
    }
    virtual_ipaddress {
        192.168.1.211
    }
}
```

Troubleshooting

If you are using a SMP server and having problems with "vrrp wdog socket" startup, try starting vrrp and the checker threads separately, e.g.:

```
keepalived --vrrp
keepalived --check
```

Links

Please view the following links for more information:

Documentation:

http://www.keepalived.org/documentation.html

http://world.anarchy.com/~peter/keepalived.conf.SYNOPSIS

(If the SYNOPSIS link is out of date, please send an email to pmueller at sidestep.com, thanks!)

IBM Redbook VRRP paper [http://www.redbooks.ibm.com/redpapers/pdfs/redp3657.pdf]

Mailing list:

http://www.keepalived.org/listes.html

Searchable, threadable mail archive

http://marc.theaimsgroup.com/?l=keepalived-devel&r=1&w=2

Chapter 16. LEAF for the pcengines WRAP

Erich Titl <eric.titl at think.ch>

Revision History
Revision 0.1 2004-08-01 eTitl
Initial document

<authorblurb>

I would like to dedicate this to the enthusiasts who made this product possible. Special thanks to Eric Spakman who was gentle enough to be convinced by my reasoning and to K.-P. Kirchdörfer who helped me to get my act together and write this little introduction. Erich Titl
</authorblurb>

The challenge

I got my hands on a nifty little SBC which had all the markings of being a perfect platform [http://www.pcengines.ch/wrap] for LEAF. Installing und running a LEAF standard distribution went without much trouble but there were a few little quirks which annoyed me.

PCengines WRAP Hardware

Pcengines WRAP is a small single board computer optimized for wireless access and network routing applications. It is built on the low power National Geode SC1100 processor and has 2 or 3 LAN sockets.

Features

- · National SC1100 CPU, 266 MHz 5x86 CPU, 16KB cache
- 2 or 3 Ethernet channels (National DP83816)
- 2 or 1 miniPCI sockets for 802.11 wireless cards and other expansion (better performance than Card-Bus, adapters should be lower cost soon)
- 64 MB SDRAM, 64 bit wide for higher memory bandwidth compared to AMD ElanSC520 based boards.
- 128 KB flash for tinyBIOS system BIOS and optional PXE boot.
- CompactFlash header for user’s operating system and application

- 12V DC supply through DC jack or passive power over LAN 1 connector
- 1 serial port (DB9 male)
- Watchdog timer built into SC1100 CPU
- LM77 thermal monitor
- Header for I2C bus (can be used for front panel interface)
- Header for LPC bus (can be used for I/O expansion)

Please refer to the above link for more details on the PCengines wrap platform.

The problem area

- At boot up the serial port spewed hundreds of messages about a jammed keyboard controller.
- The system would hang after the shutdown command.

Analysis

I quickly found that the messages and the system hang had a common reason. The board I was playing with was missing a keyboard controller. But why would the absence of a keyboard controller hang a system? Simply put, the kernel uses the keyboard controller to issue a reset to the processor. So we had 2 problems to solve

Get rid of those irritating messages at system boot, more of a cosmetic issue as the system runs fine once the keyboard init timed out.

Provide a method to overcome the katatonic state at shutdown, e.g. reboot the system.

Keyboard controller jammed messages

These messages are generated at an early kernel initialisation state when the kernel tries check and initiate the keyboard controller. They do not interfere with the normal system operation but are a nuisance. The Linux kernels up to 2.4 expect the presence of a keyboard controller and react kind of annoyed if it is missing. There is a patch by Randy Dunlap available to fix this but it would interfere with the standard hardware used on Bering boxes. Therefore this patch is *not* included in the standard distribution. Roll your own kernel if you feel the need to get rid of those messages. A copy of the patch can be found here [http://cvs.sourceforge.net/viewcvs.py/leaf/devel/etitl/kernel/kbc_option_2420.patch].

Enable reboot without use of the the keyboard controller

To enable the reboot of the system several options are available, either write a driver which would perform the necessary system related operation, or use the internal watchdog of the SC1100 processor to reset the system if the watchdog does not receive a reset signal within a predefined interval . The platform I had in mind was geared towards 24/24 service so I opted for the watchdog, especially as I could find a driver [http://www.conman.org/software/wd1100/] which handles the hardware watchdog.

The wd1100 driver

This driver enables the internal hardware watchdog timer of the sc1100 processor.

The Bering kernel has the softdog driver compiled statically into the kernel. It must be made a module in order to use the wd1100 driver.

The wd1100 driver implements the devfs interface, so it is very easy to control its behaviour through the files it presents in */proc/sys/dev/wd*. The following is needed to set the wd1100 up for automatic reboot.

- insert the wd1100 module. The driver allows the specification of the base address of the configuration block as a parameter. Normally this is only needed If the BIOS does not set the address of the configuration block to the scratch pad register. Use the gcb parameter to tell the driver where the configuration block is located.

```
insmod wd1100 [gcb=0x9000]
```

- set the wd1100 watchdog to reset when /usr/sbin/watchdog dies

```
echo 0 > /proc/sys/dev/wd/graceful
```

- set the timeout to 2 times the value of the watchdog timer interval. /usr/sbin/watchdog writes every 10 seconds a single byte to /dev/watchdog.

```
echo 20 > /proc/sys/dev/wd/timeout
```

syslinux.conf

Syslinux.conf must be set up for a serial console to monitor the system start . Modify your syslinux.conf file according to this [http://leaf.sourceforge.net/doc/guide/buconsole.html] documentation.

The solution

Bering uClibc

Bering uClibc starting at 2.0rc2 provides a kernel which modularizes softdog and includes a wd1100.o module in the distribution. Along with this comes a `/etc/modules` file which loads softdog by default, but here wd1100 can be defined as an alternative. This fits the existing model of module initialisation and makes other changes in the initialisation unnecessary. The real beauty of this is that userland does not need to be touched. If the watchdog driver is initialised as documented above then killing the watchdog program will reset the system.

The standard distribution does not include the keyboard patch. This slows down startup a little bit while the error messages are sent. A replacement kernel can be found in the cvs repository of Leaf Bering-uClibc [http://cvs.sourceforge.net/viewcvs.py/leaf/bin/bering-uclibc/packages].

Bering

Bering does not yet provide a modified kernel, you will have to roll your own using the instructions by Jacques Nilo which can be found here [http://leaf.sourceforge.net/doc/guide/bdkernel.html].

Chapter 17. Revision history

Version 0.10

Date: 2004-11-03

- New chapter for keepalived and rrdtool

Version 0.9

Date: 2004-06-03

- New chapter for dnsmasq

Version 0.8

Date: 2004-05-02

- New chapter for ip_conntrack

Version 0.7

Date: 2004-03-31

- New chapter for freenet6.lrp

Version 0.6

Date: 2004-01-01

- New chapters for pppoa/pppoe/pppoatm by E.Spakman

Version 0.5

Date: 2004-01-09

- New chapter for pcengines WRAP by E.Titl

Version 0.4

Date: 2003-12-04

- New chapter about ppp

Version 0.3

Date: 2003-10-30

- New chapter for IDE devices

Version 0.2

Date: 2003-08-17

- Changelog moved to separate chapter called "Revision history"
- Chapter "Zebra" added (E. Spakman)

Version 0.1

Date: 2003-08-11

Initial document with the following chapters:

- Structure of the document (E. de Thouars)
- Using Dropbear (M. Johnston, K.P. Kirchdoerfer, E. de Thouars)
- Configuring IPv6 (E. de Thouars)